A Supervised Learning Framework in the Context of Multiple Annotators

Julián Gil González
Andrés Marino Álvarez Meza

Facultad de Ingeniería y Arquitectura
VIE - Vicedecanatura de Investigación y Extensión
MANIZALES
2023

UNIVERSIDAD NACIONAL DE COLOMBIA

A Supervised Learning Framework in the Context of Multiple Annotators

Julián Gil González
Andrés Marino Álvarez Meza

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Bogotá, D. C., 2023

Catalogación en la publicación Universidad Nacional de Colombia

Gil González, Julián, 1991-
A Supervised Learning Framework in the Context of Multiple Annotators / Julián Gil González, Andrés Marino Álvarez Meza. — Primera edición. — Bogotá : Universidad Nacional de Colombia. Editorial Universidad Nacional de Colombia; Manizales : Universidad Nacional de Colombia. Vicedecanatura de Investigación y Extensión. Facultad de Ingeniería y Arquitectura, 2023
1 CD-ROM (118, páginas) : ilustraciones (principalmente a color), diagramas

Incluye referencias bibliográficas e índice analítico

ISBN 978-958-505-369-4 (e-book)

1. Aprendizaje supervisado (Inteligencia artificial) 2. Aprendizaje automático (Inteligencia artificial) 3. Redes neuronales (Computadores) 4. Procesos de Gauss 5. Ciencia de datos I. Álvarez Meza, Andrés Marino, 1988- II. Título

CDD-23 006.31 / 2023

Editorial Universidad Nacional de Colombia
Alberto Amaya Calderón
Director

Comité editorial
Alberto Amaya Calderón
Ana Patricia Noguera de Echeverry
Fabio Andrés Pavas Martínez
Veronique Claudine Bellanger
Fredy Fernando Chaparro Sanabria
Jairo Iván Peña Ayazo
Pedro Nel Benjumea Hernández

Primera edición, 2023
ISBN (digital): 978-958-505-369-4

Edición
Editorial Universidad Nacional de Colombia
direditorial@unal.edu.co
www.editorial.unal.edu.co

Equipo editorial
Coordinación editorial: John Fredy Guzmán
Corrección de estilo: Rosa Isabel González
Pauta gráfica: Juan Carlos Villamil Navarro
Diagramación: Andrés Merino Álvarez Meza
Diseño de cubierta: Juan Carlos Villamil Navarro

Hecho en Bogotá, D. C., Colombia, 2023

About the authors

Julián Gil González. Received a bachelor's degree in Electronic Engineering, an M.Sc. in Electrical Engineering, and a Ph.D. in Engineering from the Universidad Tecnológica de Pereira (Pereira, Colombia), in 2014, 2016, and 2021, respectively. His research interests include probabilistic models for machine learning, learning from crowds, and Bayesian inference.

Andrés Marino Álvarez Meza. Received a bachelor's degree in Electronic Engineering, an M.Sc. degree in Engineering, and a Ph.D. degree in Automatics from the Universidad Nacional de Colombia - Sede Manizales (Colombia), in 2010, 2012, and 2016, respectively. He is currently a Professor at the Department of Electrical, Electronic, and Computation Engineering, Universidad Nacional de Colombia - Sede Manizales. His research interests include machine learning and signal processing.

Contents

1 Preliminaries **2**

1.1 Motivation . . . 2

1.2 Problem Statement . . . 3

1.3 Mathematical Preliminaries . . . 5

1.3.1 Methods for Supervised Learning . . . 5

1.3.2 Learning from Multiple Annotators . . . 10

1.4 Literature Review on Supervised Learning from Multiple Annotators . . . 10

1.5 Objectives . . . 16

1.5.1 General Objective . . . 16

1.5.2 Specific Objectives . . . 16

1.6 Outline and Contributions . . . 17

1.6.1 Kernel Alignment-Based Annotator Relevance Analysis (KAAR) . . . 17

1.6.2 Localized Kernel Alignment-Based Annotator Relevance Analysis (LKAAR) . . . 18

1.6.3 Regularized Chained Deep Neural Network for Multiple Annotators (RCDNN) 19

1.6.4 Chained Gaussian Processes for Multiple Annotators (CGPMA) and Correlated Chained Gaussian Processes for Multiple Annotators (CCGPMA) . . . 21

1.6.5 Book Structure . . . 22

2 Kernel Alignment-Based Annotator Relevance Analysis **24**

2.1 Centered Kernel Alignment Fundamentals . . . 24

2.2 Kernel Alignment-Based Annotator Relevance Analysis . . . 25

2.2.1 KAAR for Classification and Regression . . . 27

2.3 Experimental Set-Up . . . 27

2.3.1 Classification . . . 27

2.3.2 Regression . . . 30

2.4 Results and Discussion . . . 32

2.4.1 Classification . . . 32

2.4.2 Regression . . . 35

2.5 Summary . . . 39

3 Localized Kernel Alignment-Based Annotator Relevance Analysis **40**
3.1 Localized Kernel Alignment Fundamentals . . . 40
3.2 Localized Kernel Alignment-Based Annotator Relevance Analysis . . . 41
3.2.1 LKAAR for Classification and Regression . . . 43
3.3 Experimental Set-Up . . . 43
3.3.1 Classification . . . 43
3.3.2 Regression . . . 45
3.4 Results and Discussion . . . 46
3.4.1 Classification . . . 46
3.4.2 Regression . . . 55
3.5 Summary . . . 57

4 Regularized Chained Deep Neural Network for Multiple Annotators **59**
4.1 Chained Deep Neural Network . . . 59
4.2 Regularized Chained Deep Neural Network for Classification with Multiple Annotators 60
4.3 Experimental Set-Up . . . 61
4.3.1 Tested Datasets . . . 61
4.3.2 Provided and Simulated Annotations . . . 62
4.3.3 Method Comparison and Quality Assessment . . . 62
4.3.4 RCDNN Detailed Architecture and Training . . . 63
4.4 Results and Discussion . . . 64
4.5 Summary . . . 71

5 Correlated Chained Gaussian Processes for Multiple Annotators **73**
5.1 Chained Gaussian Processes . . . 73
5.1.1 Correlated Chained Gaussian Processes . . . 75
5.2 Correlated Chained GP for Multiple Annotators-CCGPMA . . . 77
5.2.1 Classification . . . 77
5.2.2 Regression . . . 78
5.3 Experimental Set-Up . . . 79
5.3.1 Classification . . . 79
5.3.2 Regression . . . 81
5.4 Results and Discussion . . . 82
5.4.1 Classification . . . 82
5.4.2 Regression . . . 88
5.5 Summary . . . 94

6 Final Remarks **95**
6.1 Conclusions . . . 95

6.2 Future Work 98
6.3 Repositories 100

Bibliography **101**

Appendices **106**

Appendix A CCGPMA Supplementary Material **107**
A.1 Derivation of CCGPMA Lower Bounds 107
A.1.1 Gradients w.r.t. the Variational Parameters 108
A.2 Likelihood Functions 109
A.2.1 Multiclass Classification with Multiple Annotators 109
A.2.2 Gaussian Distribution for Regression with Multiple Annotators 114

Alphabetical Index **118**

List of plots

Figure 1.1 – Samples from a GP with SE covariance function over a 1-D problem with $s = 1$ and $l = 0.2$ 8

Figure 1.2 – Graphical comparison between datasets for typical supervised learning settings and datasets from multiple annotators 11

Figure 1.3 – Graphical model for the approach proposed in [1]. $\boldsymbol{x}_n$, y_n, and $y_n^{(r)}$ are, respectively, the input, the ground truth, and the label given by the r-th annotator. Shaded nodes represent observed values, while unshaded nodes indicate latent variables 12

Figure 1.4 – The graphical model used to code the labelers' expertise as a function of the input space. $\boldsymbol{x}_n$, y_n, and $y_n^{(r)}$ are the input, the ground truth, and the label given by the r-th annotator, respectively. Shaded nodes represent observed values, while unshaded nodes indicate latent variables 13

Figure 1.5 – Relevant state-of-the-art works for supervised learning with multiple annotators 16

Figure 1.6 – Relationship between the aims of this work and the developed methodologies. KAAR: Kernel alignment-based annotators relevance analysis. LKAAR: Localized kernel alignment-based annotator relevance analysis, RCDNN: Regularized chained deep neural network for Multiple Annotators, CGPMA: Chained Gaussian Processes for multiple annotators, and CCGPMA: Correlated chained Gaussian Processes for multiple annotators 17

Figure 1.7 – Kernel alignment-based annotator relevance analysis – (KAAR) pipeline. CKA stands for Centered Kernel Alignment. $\mathcal{X}$ indicates the input space, and each $\mathcal{Y}_r$ represents the output space for the r-th annotator. Moreover, $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_R]^\top$ is a vector containing the annotators' relevance parameters. Finally, the supervised learning mapping function f is computed as a convex combination of functions g_r that are trained from the r-th annotator's data 19

Figure 1.8 – Localized kernel alignment-based annotator relevance analysis – (KAAR) pipeline. CKA stands for Centered Kernel Alignment. $\mathcal{X}$ indicates the input space, and each $\mathcal{Y}_r$ represents the output space for the r-th annotator. Moreover, $[q_1(\boldsymbol{x}_n), \ldots, q_R(\boldsymbol{x}_n)]^\top$ is a vector containing the annotators' relevance parameters for the n-th input sample. Finally, the supervised learning mapping function f is computed as a convex combination of functions g_r that are trained from the r-th annotator's data 20

Figure 1.9 – Regularized chained Deep Neural Network scheme. $\boldsymbol{x}$ is an input vector, $\varrho_o(\cdot)$ is the o-th layer of a DNN, $f_j(\boldsymbol{x})$ is the j-th output neuron for the input $\boldsymbol{x}$. Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood modeled as a function of the input space, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$ . . . 21

Figure 1.10 – Chained Gaussian processes. Each $f_j(\boldsymbol{x})$ is a latent function that follows a GP, Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$ 22

Figure 1.11 – Correlated chained Gaussian processes. Each $f_j(\boldsymbol{x})$ is a latent function computed as a linear combination of functions $\{\mu_q(\boldsymbol{x})\}_{q=1}^{Q}$ that follows a GP. Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$ 22

Figure 2.1 – Iris dataset illustrative results. On the first row, from left to right, we show the similarities among samples according to the true labels and the input features. On the second and third rows, from left to right, we expose the kernels computed from the annotators' labels concerning each simulation method (see Table 2.5). Vertical and horizontal axes display the sample index, sorted by the value of the ground truth labels 33

Figure 2.2 – Annotators' interdependencies. In the left column, from top to bottom, we show the Pearson coefficient computed over the labels from multiple annotators. On the right, we present the annotators' dependencies estimated with our KAAR 34

Figure 2.3 – Regression illustrative results. In the first column, from top to bottom, we show the regression results for each annotator. In the second column, from top to bottom, we exhibit the result for GPR-GT, the result for our GPR-KAAR; finally, in the last row, we show the Pearson coefficient computed over the labels from multiple annotators and the dependencies estimated with our KAAR 37

Figure 3.1 – Fully synthetic dataset results. The first column (top to bottom): relevance values $q_r^2(\boldsymbol{x}_n)$. The second column (top to bottom): decision boundaries produced by a GPC trained over each annotator dataset. The third column (top to bottom): decision boundaries generated by the GPC-GOLD (gold standard) and the LKAAR. Also, it displays the dependencies among the annotators estimated by LKAAR (from the input samples) vs. the Pearson correlation coefficients (absolute value) from the labelers' annotations 48

Figure 3.2 – Annotator dependencies analysis for the Voice dataset (Scale B). First row (left to right): t-sne-based 2D projections holding the annotators' labels. Red circles: positive class, blue crosses: negative class. Second row (left to right): dependencies among annotators estimated by LKAAR (from the input samples) vs. the Pearson correlation coefficients (absolute value) from the experts' labels 52

Figure 3.3 – Number of labeled instances per annotator 54

Figure 3.4 – Generalization performance in terms of AUC as a function of the number of annotators 54

Figure 3.5 – Regression illustrative results. In the first column, from top to bottom, we show the regression results g_r for each annotator. On the second column, from top to bottom, we present the LKAAR-based relevance q_r. In the third column, we exhibit the result for GPR-GOLD and the result for our GPR-LKAAR; finally, in the last row, we show the Pearson coefficient computed over the labels from multiple annotators and the dependencies estimated with our LKAAR 56

Figure 4.1 – RCDNN architecture details. ϱ_s stands for dense layer. ϱ_1 holds a linear activation, ϱ_2 includes a tanh-based activation, and ϱ_3 and ϱ_4 output the hidden ground truth label and the annotator's reliability fixing a softmax and a sigmoid activation, respectively 64

Figure 4.2 – RCDNN's decision boundaries for the 2D-PCA Iris dataset (synthetic scenario). AUC$= 0.9837$. The point's color stands for the Iris dataset classes. PCA_1 and PCA_2 stand for the first and second PCA-based projections . . . 65

Figure 4.3 – RCDNN-based annotators' performance (reliability) estimation for the synthetic experiments (2D PCA Iris data). In the first column (from top to bottom), the simulated accuracy for each annotator is presented based on equation (4.10). The second column shows (from top to bottom) the estimated annotators' reliability (λ_r) 66

Figure 4.4 – Target and estimated annotators' dependencies for the synthetic 2D PCA Iris dataset. On the left, the Pearson correlation coefficients (absolute value) from simulated accuracies (experts reliability) in matrix $\boldsymbol{P}$ of equation (4.10) are shown. On the right, the dependencies among the annotators estimated from the RCDNN ϱ_4 layer's weights are displayed 67

Figure 4.5 – Receiver Operating Characteristic (ROC) plot for the annotators simulated within the spammers and malicious scenario. Blue dots indicate the basis annotators. Red dots show extra annotators with parameters $R_e = 65$ and $p_r = 0.5$. Green dots specify extra labelers with $R_e = 20$ and $p_r = 0.6$. We notice that annotators located in the dashed line vicinity are considered Spammers. Similarly, labelers above in the dashed line are regarded as good annotators; conversely, labelers located below such a line are malicious annotators 70

Figure 4.6 – MA-LFC, LKAAR-GPC, and RCDNN performance (AUC) as a function of the number of labelers (spammers and malicious annotators) 71

Figure 5.1 – Fully synthetic dataset results. The PLP is shown, comparing the prediction of our CCGPMA-C ($AUC = 1$) and CCGPMA-C ($AUC = 0.9999$) against: the theoretical upper bound GPC-GOLD ($AUC = 1.0$), the lower bound GPC-MV($AUC = 0.9809$), and the state-of-the-art approaches MA-LFC-C ($AUC = 0.9993$), MA-DGRL ($AUC = 0.9999$), MA-GPC ($AUC = 0.9977$), MA-GPCV ($AUC = 0.9515$), MA-DL-MW ($AUC = 0.9989$), MA-DL-VW ($AUC = 0.9972$), MA-DL-VW+B ($AUC = 0.9994$), KAAR (0.9099). The shaded region in GPC-MV, CGPMA-C, and CCGPMA-C indicates the area enclosed by the mean $\pm$ two standard deviations. There is no shaded region for approaches lacking prediction uncertainty 84

Figure 5.2 – Fully synthetic data reliability results. From top to bottom, the first column exposes the true reliabilities (λ_r). The subsequent columns present the estimation of the reliabilities performed by state-of-the-art models, where the correct values are provided in dashed lines. The shaded region in CGPMA-C and CCGPMA-C indicates the area enclosed by the mean $\pm$ two standard deviations. Also, the accuracy (Acc) is provided 85

Figure 5.3 – Fully synthetic dataset results. We compare the prediction of our CCGPMA-R ($R^2 = 0.9438$), and CGPMA-R ($R^2 = 0.9280$) with the theoretical upper bound GPR-GOLD ($R^2 = 0.9843$) and lower bound GPR-Av ($R^2 = 0.8718$), and state-of-the-art approaches, MA-LFCR ($R^2 = -0.0245$), MA-GPR ($R^2 = 0.9208$), MA-DL-B ($R^2 = 0.7020$), MA-DL-S ($R^2 = 0.6559$), MA-DL-B+S ($R^2 = 0.5997$). Note that we provided the Gold Standard in dashed lines. The shaded region in GPR-Av, MA-GPR, CGPMA-R, and CCGPMA-R indicate the area enclosed by the mean plus or minus two standard deviations. We remark that there is no shaded region for MA-LFCR and DLMA since they do not provide information about the prediction uncertainty 89

Figure 5.4 – Estimated values of error variance for the five annotators in the *fully synthetic* experiment. In the first column, from top to bottom, we expose the error variances used to simulate the labels from each annotator. Furthermore, the subsequent columns from top to bottom present the estimation of such error variances performed by state-of-the-art models that include these parameters in their formulation; moreover, the true error variances are provided in dashed lines. The shaded region in CGPMA-R and CCGPMA-R indicates the area enclosed by the mean plus or minus two standard deviations. We remark that there is no shaded region for MA-LFCR and MA-GPR since these approaches perform a fixed-point estimation for the annotators' parameters. Finally, we remark that the R^2 score between the true and estimated error variances is provided 90

Figure 6.1 – Malicious annotators. On the left, we show the Receiver's Operating Characteristic (ROC) plot for the simulated annotators. Red dots indicate the basis annotators. Green dots show malicious annotators. On the right, we present the performance (AUC) of DGRL, MA-GPC, MA-DL-VW+B, KAAR, LKAAR, RCDNN, CGPMA, and CCGPMA as a function of the number of malicious annotators 98

List of tables

2.1 Datasets description 29
2.2 A brief overview of the state-of-the-art methods tested 30
2.3 Datasets for regression 31
2.4 A brief overview of state-of-the-art methods tested for regression tasks 32
2.5 KAAR-based annotator coding results for the iris dataset 33
2.6 UCI repository classification results 35
2.7 Real annotators datasets results 36
2.8 KAAR-based annotator coding results in the *fully synthetic* dataset 36
2.9 Regression results in terms of R^2 score over *semi-synthetic datasets* 38
2.10 Regression results in terms of R^2 score over the *fully real dataset* 38

3.1 A brief overview of the state-of-the-art methods tested 45
3.2 A brief overview of state-of-the-art methods tested for regression tasks 46
3.3 UCI repository classification results 49
3.4 Fully real datasets results 51
3.5 Regression results in terms of R^2 score over *semi-synthetic datasets* 56
3.6 Regression results in terms of R^2 score over the *fully real dataset* 57

4.1 A short overview of the tested state-of-the-art approaches 63
4.2 Semi-synthetic datasets results 68
4.3 Fully real-world datasets results 69

5.1 A brief overview of the state-of-the-art methods tested 81
5.2 Semi-synthetic classification results 86
5.3 AUC classification results for the fully real datasets 88
5.4 Semi-synthetic regression results 91
5.5 Regression results regarding R^2 score over the *fully real dataset* 93

6.1 AUC classification results for the semi-synthetic and fully real datasets 97

Chapter 1. Preliminaries

1.1 Motivation

The accelerated use of information technologies in different domains is changing how datasets are built. For instance, dedicated crowdsourcing platforms like Amazon Mechanical Turk (AMT), LabelMe, and Crowdflower allow extracting the 'wisdom of crowds' [2] to obtain large datasets labeled by multiple annotations. However, the concept of crowdsourcing goes beyond dedicated platforms; for example, through its social nature, the web collects a large volume of labeled datasets, where such labels commonly correspond to web-based ratings (products rating, translation rating, product tag). The attractiveness of crowdsourcing lies in the possibility of getting suitable quality labels at a low-cost [3, 4].

The aforementioned poses a new challenge in the supervised learning context. Instead of having datasets labeled by one source (which is supposed to be an expert who provided the absolute gold standard), we have datasets labeled by multiple annotators with different and unknown expertise [5]. Hence, typical supervised learning algorithms need to be adapted to face multi-labeler datasets.

Besides, in a local context, the research group in automatics (RGA) from Universidad Tecnológica de Pereira and the Grupo de control y procesamiento digital de señales from Universidad Nacional de Colombia (Sede Manizales) have successfully developed several research works related to the analysis of different medical images (ultrasound, X-rays, MRI, DMRI) aiming to support the identification of nerves structures and the diagnosis of various pathologies, including Parkinson, breast, and brain cancer [6–8]. Most of these works include a supervised learning step, where it is necessary to hire experts to obtain the required labels for training the learning algorithms. However, experts are scarce, and their time is costly; moreover, there exists disagreement among the annotations given by the multiple annotators, especially if they have different levels of expertise.

Accordingly, from the global and local scenarios, it is necessary to continue developing methodologies that allow facing supervised learning problems in the context of multiple annotators. This book presents the latest advancements in supervised learning from multiple annotators, drawing from the doctoral thesis of Dr. Julián Gil Gonzalez and the collective research efforts of the previously mentioned groups.

1.2 Problem Statement

The main aim of a supervised learning task is to learn a function that maps from the input features to the output space, which is estimated by using a training set, where it is supposed that an expert provides the actual label (termed gold standard) for each instance [9]. Nonetheless, in many real-world applications, such a gold standard is unavailable since the experts are scarce, their time is expensive, and the labeling problem is tedious and time-consuming [10, 11]. Similarly, other labeling problems correspond to a subjective evaluation (*e.g.*, sentimental analysis, products rating); thus, the gold standard needs to be clarified [12]. Instead of the gold standard, we may have access to several noisy annotations provided by R heterogeneous annotators (also named as sources or labelers), where each source gives its version of the unknown gold standard [13]. Those annotations can be collected in several ways. For instance, the model proposed in [1], the sources correspond to physicians who make a diagnosis about the presence of cancer using medical images. Likewise, in the approach proposed by the authors in [10], the labels are provided by algorithms that are used to measure the QT interval in an electrocardiogram signal (the QT interval is a measurement used to assess some electrical properties of the heart).

Accordingly, conversely to typical supervised learning settings, in multi-labeler scenarios, each instance is linked with multiple annotations provided by multiple annotators. Commonly, not all the labelers give an output for each input in the dataset. Thereby, it is not straightforward to apply traditional supervised learning algorithms in the presence of data from multiple annotators [14]. In this sense, *learning from crowds* has been introduced as a general framework from two main perspectives: to fit the labels from multiple annotators or to adapt the supervised learning algorithms [15].

The first approach is known in the literature as "label aggregation" or "truth inference," comprising the computation of a single hard label per sample to estimate of the ground truth. The hard labels are then used to feed a standard supervised learning algorithm [16]. The straightforward method is the so-called majority voting–(MV), which has been used in different multi-labeler problems due to its simplicity [17]. Still, MV assumes homogeneity in annotators' reliability, which is hardly feasible in real applications, *e.g.*, experts vs. spammers. Furthermore, the consensus is profoundly impacted by incorrect labels and outliers [13]. Conversely, more elaborated models have been considered to improve the estimation of the correct tag through the well-known Expectation-Maximization–(EM) framework and by facing the imbalanced labeling issue [17, 18].

The second approach jointly trains the supervised learning algorithm and models the annotators' behavior. It has been shown that such strategies lead to better performance than those belonging to label aggregation. Thus, the features used to train the learning algorithm provide valuable information to puzzle the ground truth [19]. The most representative work in this area is exposed in [1], which offers an EM-based framework to learn the parameters of a logistic re-

gression classifier and model the annotators' behavior by computing their sensitivities and specificities. Such a technique has inspired several models in the context of multi-labeler scenarios, including binary classification [19, 20] , multi-class discrimination [16, 21], regression [22, 23], and sequence labeling [23, 24]. Furthermore, some works have addressed the multi-labeler problem using deep learning approaches, typically including an extra layer that codes the annotators' information [3, 25, 26].

According to the above, among the works developed in this area, we recognize two main groups: the approaches based on frequentist models–(FMs) and the schemes based on probabilistic or Bayesian models–(BMs). Furthermore, we note that most works on learning from crowds are mainly based on two assumptions: *i) The performance of the annotators does not depend on the feature space* and *ii) independence among the annotators*; however, these assumptions are not valid in practice. Hence, the primary motivation of this proposal is to address these two problems.

The performance of the annotators does not depend on the feature space. In most multi-labeler approaches, it is necessary to estimate the performance of the annotators, which is usually measured in terms of accuracy [27], or sensitivity and specificity [1, 3, 19] in classification settings; similarly, in regression approaches, such performance is measured in terms of the error variance [22]. However, a restriction commonly seen in these algorithms is that they assume that the performance is consistent across the input space. This assumption is an impractical restriction since the expertise of the annotators may vary depending on the instance they label [28, 29]. For example, if we consider online annotators assessing some documents, they may have different labeling accuracy. Such differences may rely on whether they are more familiar with specific topics related to studied documents [30].

Independence among the annotators. Another assumption commonly seen is to consider independence among the annotators. This assumption is used to reduce the complexity of the model [31, 32] or based on the fact that it is plausible to guarantee that each labeler performs the annotation process individually [33]. Nevertheless, this assumption cannot fit most real-world applications. For example, if the sources are humans, the independence assumption is hardly feasible because knowledge is a social construction; hence, people's decisions will be correlated since they share information, communicate with each other, or because they belong to a particular school of thought [34, 35]. On the other hand, if we consider that the sources are algorithms, where some of them are based on the same math principle, there likely exists a correlation among their labels [10]. Accordingly, the relaxation of this restriction could be used to improve the ground truth estimation [36].

Therefore, some problems related to supervised learning with multiple annotators still need to be solved. For this reason, the main objective of this work is to develop multiple annotators'

models based on FMs and BMs aiming to code the labelers' performance as a function of the input space and considering dependencies among them to improve the multi-labeler representation the ability to classify and regress tasks.

1.3 Mathematical Preliminaries

In this section, we introduce the mathematical formulation for the problem of learning from multiple annotators. First, in Section 1.3.1, we recall the mathematical formulation of a traditional supervised learning scenario. We analyze two perspectives to face such scenarios: The frequentist and the Bayesian point of view. Finally, Section 1.3.2 describes the composition of a dataset from multiple labelers and the main aims of algorithms dealing with this kind of data.

1.3.1 Methods for Supervised Learning

A classical supervised learning problem comprises the estimation of a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are, respectively, the input space and the output space [9]. Such a function f is computed from a training set $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$, where $\boldsymbol{X} = \left\{\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^P\right\}_{n=1}^{N}$ and $\boldsymbol{y} = \{y_n \in \mathcal{Y}\}_{n=1}^{N}$. Depending on the output space $\mathcal{Y}$ nature, we recognize different supervised learning settings: *i) binary classification* holding $\mathcal{Y} \in \{-1, 1\}$ or $\mathcal{Y} \in \{0, 1\}$; *ii) multi-class classification*, where $\mathcal{Y} \in \{1, \ldots, K\}$, being K the number of classes; and *iii) regression* where $\mathcal{Y} \in \mathbb{R}$ [1].

Notice that in most supervised learning cases, we are not interested in knowing the exact form f, but we are focused on computing the output in a specific value $f(\boldsymbol{x}_*)$, where $\boldsymbol{x}_* \in \mathbb{R}^P$ [37]. One of the most basic estimators is the well-known generalized linear estimator given by:

$$f(\boldsymbol{x}_*) = \vartheta_0 + \sum_{d=1}^{D} \vartheta_d \xi_d(\boldsymbol{x}_*), \tag{1.1}$$

where $\{\vartheta_d\}_{d=0}^{D}$ are the model parameters, and $\xi_d : \mathbb{R}^P \to \mathbb{R}$ is the d-th component of a pre-selected set of non-linear functions, named basis functions, aiming to deal with non-linearities in the data's structure [38]. The main advantage of these type of estimators is that the related models remain linear regarding the set of parameters $\{\vartheta_d\}_{d=0}^{D}$. Nevertheless, because this estimator is parametric, for high-dimensional spaces, the number of basis functions D has to be large to obtain a proper performance, leading to overfitting [39]. Such drawback has been addressed in the literature; for example, Support Vector Machines–(SVMs) define a set of basis functions centered in the training data. Then, a subset of them is selected during the training, where the number of selected functions is generally smaller than the training set's size. Another alternative propound to use parametric forms for the basis functions (the number of functions is given in advance), in which the parameters are estimated during the training stage. The most successful approach in this context is the well-known artificial neural network–(ANN) [9].

On the other hand, in contrast to previous approaches, which are parametrized in terms of a set of basis functions, it is possible to introduce a symmetric positive bivariate function, termed *kernel*, which enables to build non-parametric estimators less prone to overfitting. Thus, a way to estimate $f(\boldsymbol{x}_*)$ is given as follows [40]:

$$f(\boldsymbol{x}_*) = \boldsymbol{k}_*^\top \left(\boldsymbol{K} + \psi N \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{1.2}$$

where $\boldsymbol{k}_* = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}_*), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x}_*)]^\top \in \mathbb{R}^{N\times 1}$, and $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel function. Besides, $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is formed by the evaluation of κ over the input set $\boldsymbol{X}$, and $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ is the identity matrix. Finally, $\psi \in \mathbb{R}^+$ is a regularization hyper-parameter, which will be discussed in the following sections. Hence, such estimator in equation (1.2) can be derived from two different but related perspectives: a frequentist (regularization) and a Bayesian perspective.

A Frequentist Perspective

From a frequentist perspective (regularization), the aim is to suppose the function of interest to belong to a reproducing kernel Hilbert space–(RKHS) $\mathcal{H}_\kappa$, which is generated by the chosen kernel κ; thereby, we have $f \in \mathcal{H}_\kappa$. Accordingly, the estimator can be inferred by minimizing the following regularized functional

$$\hat{f} = \underset{f\in\mathcal{H}_\kappa}{\arg\min} \frac{1}{N}\sum_{n=1}^{N} \left(f(\boldsymbol{x}_n) - y_n\right)^2 + \psi ||f||^2_{\mathcal{H}_\kappa}, \tag{1.3}$$

where $||\cdot||_{\mathcal{H}_\kappa}$ is the norm in $\mathcal{H}_\kappa$. Besides, we notice that the first term in equation (1.3) corresponds to the well-known empirical risk, which particularly is computed as the sum of the squared errors [40]. Such a term penalizes the variation between predicted outputs $f(\boldsymbol{x}_n)$ and the corresponding true value y_n. The second term in equation (1.3) represents the regularization expression that controls the estimator's smoothness to avoid overfitting; hence, the larger value, the smoother the estimation $\hat{f}$. The regularization concept in a RKHS plays a key role in the machine learning area, so it is necessary to review some notions about RKHS [41]. A RKHS is a Hilbert space of functions defined by a reproducing kernel κ, a symmetric positive definite function. In that sense, given a kernel function κ a RKHS is generated in such a way that the function $\kappa(\boldsymbol{x}, \cdot)$ is linked to $\mathcal{H}_\kappa$ for all $\boldsymbol{x} \in \mathcal{X}$, and [42]

$$f(\boldsymbol{x}) = \langle f, \kappa(\boldsymbol{x}, \cdot)\rangle_{\mathcal{H}_\kappa}; \quad \forall f \in \mathcal{H}_\kappa,$$

where $\langle\cdot,\cdot\rangle_{\mathcal{H}_\kappa}$ is the inner product in $\mathcal{H}_\kappa$. The above expression corresponds to the *reproducing* property. Further, we recognize two additional properties, essential for the regularization perspective. First, the *representer* theorem indicates that in an RKHS the functions are built as a linear combination of the kernel in given points [43]

$$f(\boldsymbol{x}) = \sum_{n=1}^{N} \alpha_n \kappa(\boldsymbol{x}, \boldsymbol{x}_n) = \boldsymbol{k}^\top \boldsymbol{\alpha}, \tag{1.4}$$

where $\boldsymbol{k} = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{x}_N, \boldsymbol{x})]^\top \in \mathbb{R}^{N\times 1}$ and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^\top \in \mathbb{R}^{N\times 1}$ are the combination parameters. Second, the norm of a function in a given RKHS can be written as:

$$||f||_{\mathcal{H}_\kappa} = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}. \tag{1.5}$$

Then, replacing equations (1.4) and (1.5) in equation (1.3), the optimization problem becomes:

$$\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}} \frac{1}{N} \left(\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{K}\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{y} + \boldsymbol{y}^\top \boldsymbol{y} \right) + \psi \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha}. \tag{1.6}$$

We remark that the latter objective function is convex with respect to $\boldsymbol{\alpha}$; thereby differentiating such an objective function with respect to $\boldsymbol{\alpha}$, and equaling zero, the minimizer for equation (1.6) yields:

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{K} + \psi N \boldsymbol{I})^{-1} \boldsymbol{y}.$$

Accordingly, the output estimation for a new instance $f(\boldsymbol{x}_*)$ is given as follows

$$f(\boldsymbol{x}_*) = \sum_{n=1}^{N} \alpha_n \kappa(\boldsymbol{x}_*, \boldsymbol{x}_n) = \boldsymbol{k}_*^\top \hat{\boldsymbol{\alpha}}.$$

A Bayesian Perspective

On the other hand, another perspective exists to solve supervised learning problems, called Bayesian. We identify controversy in the exact definition of Bayesian approaches; for instance, [9] establishes that the Bayesian formulation stands for considering f to be random; nevertheless, [44] sets that Bayesian estimation can also be used for deterministic computation of f. In this book, we follow the definition in [9]; thus, we consider only the models that fix f to be random (or the model parameters if it is the case) as a Bayesian treatment.

For specific demonstration, we use the well-known Gaussian processes–(GPs), one of the most common Bayesian approaches in the supervised learning context. Namely, a GP is a random process where any finite set of samples follows a Gaussian distribution [37]. Typically, a GP is employed as a prior over functions. Hence, let be $f(\boldsymbol{x})$ a function that follows a GP, $f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), \kappa_f(\boldsymbol{x}, \boldsymbol{x'}))$, where $m(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x})]$ is the mean function (usually $m(\boldsymbol{x}) = 0$), and $\kappa_f(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))]$ is the covariance function with $\kappa_f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ being a given kernel function ($\boldsymbol{x}' \in \mathcal{X}$).

If we consider the finite set of inputs in $\boldsymbol{X}$, then $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ is drawn for a multivariate Gaussian distribution $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{f}\boldsymbol{f}})$, where $\boldsymbol{K}_{\boldsymbol{f}\boldsymbol{f}} \in \mathbb{R}^{N\times N}$ is the covariance matrix formed by the evaluation of κ_f over the input set $\boldsymbol{X}$.

From a Bayesian point of view, the GPs priors represent our beliefs about the properties of the function we are modeling [40]. Let us consider the squared exponential–(SE) kernel, given as:

$$\kappa_{\text{SE}}(\boldsymbol{x}, \boldsymbol{x}') = s^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2l^2}\right),$$

with the parameter $l \in \mathbb{R}^+$ defining the characteristic length-scale and $s \in \mathbb{R}^+$ specifying an output-scale amplitude. We notice in Figure 1.1 that a GP, prior to using a SE kernel function, prefers smooth functions [45]. Then, the beliefs captured by the GP prior are updated in the presence of data through the *likelihood* function. This leads to an updated distribution, named the *posterior distribution* that can be used to make predictions over new samples $\boldsymbol{x}_*$.

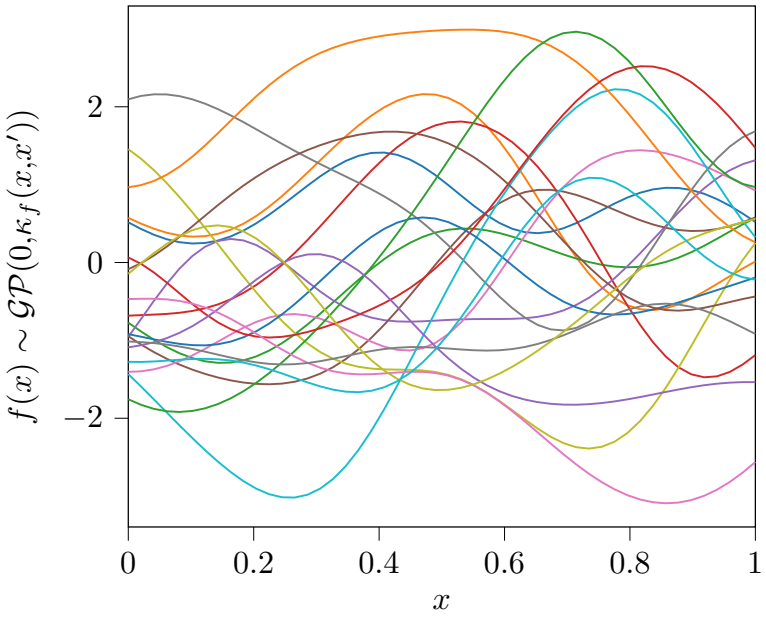

Figure 1.1 Samples from a GP with SE covariance function over a 1-D problem with $s = 1$ and $l = 0.2$

In a regression setting, the likelihood function is usually Gaussian and codes the linear relation between the observations and a given model corrupted with zero mean Gaussian noise,

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|f(\boldsymbol{x}_n), \sigma^2),$$

where $\boldsymbol{f} = [f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$, and $\sigma^2 \in \mathbb{R}^+$ is the noise variance. Notice that the likelihood function factorizes over the data points, which assumes that the noise is independent and identically distributed. In this case, due to a Gaussian likelihood, the posterior distribution has an analytic solution. Given a new test point $\boldsymbol{x}_*$ and the training data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$, the predicted distribution is computed as:

$$p(f(\boldsymbol{x}_*)|\mathcal{D}, \boldsymbol{x}_*, \boldsymbol{\phi}) = \mathcal{N}(\bar{f}(\boldsymbol{x}_*), k(\boldsymbol{x}_*, \boldsymbol{x}_*)), \tag{1.7}$$

where $\boldsymbol{\phi}$ indicates the model hyper-parameters, including σ^2, and the hyper-parameters related

to the kernel function. Moreover, we have

$$\bar{f}(\boldsymbol{x}_*) = \boldsymbol{k}_{\boldsymbol{f}f_*}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 N \boldsymbol{I}\right)^{-1} \boldsymbol{y}, \tag{1.8}$$

$$k(\boldsymbol{x}_*, \boldsymbol{x}_*) = k_{f_*f_*} - \boldsymbol{k}_{\boldsymbol{f}f_*}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_{\boldsymbol{f}f_*}, \tag{1.9}$$

where $\boldsymbol{k}_{\boldsymbol{f}f_*} \in \mathbb{R}^N$ is the cross-covariance function between $\boldsymbol{f}$ and $f_* = f(\boldsymbol{x}_*)$; besides, $k_{f_*f_*} \in \mathbb{R}$ is the covariance function for f_*. Conversely, the model hyper-parameters $\boldsymbol{\phi}$ are estimated by minimizing the minus logarithm of the marginalized likelihood function, as follows

$$\hat{\boldsymbol{\phi}} = \arg\min_{\boldsymbol{\phi}} \left(-\log p(\boldsymbol{y}|\boldsymbol{X})\right) = \frac{1}{2}\boldsymbol{y}^\top \left(\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} + \frac{1}{2}\log\left|\boldsymbol{K}_{\boldsymbol{ff}} + \sigma^2 \boldsymbol{I}\right|$$

where $|\cdot|$ indicates the determinant of a matrix.

Connections between Frequentist and Bayesian Points of View

In general terms, the dimension of an RKHS can be infinity [40]. However, if we limit the analysis to finite-dimensional RKHS, it is possible to determine that every RKHS can be described in terms of a feature map $\Phi : \mathcal{X} \to \mathbb{R}^D$, such that

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \Phi_d(\boldsymbol{x}) \Phi_d(\boldsymbol{x}'). \tag{1.10}$$

Moreover, the functions in an RKHS with a kernel κ are formulated based on a set of parameters $\boldsymbol{\varpi} = [\varpi_1, \ldots, \varpi_D]^\top \in \mathbb{R}^D$ as follows

$$f(\boldsymbol{x}) = \sum_{j=1}^{D} \varpi_d \Phi_d(\boldsymbol{x}) = \langle \boldsymbol{\varpi}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle, \quad \text{with} \quad ||f_{\boldsymbol{\varpi}}||_{\mathcal{H}_\kappa} = ||\boldsymbol{\varpi}||_2, \tag{1.11}$$

where $\langle \cdot, \cdot \rangle$ and $||\cdot||_2$ are the Euclidean inner product and norm, respectively. Besides, $\boldsymbol{\Phi}(\boldsymbol{x}) = [\Phi_1(\boldsymbol{x}), \ldots, \Phi_D(\boldsymbol{x})]^\top \in \mathbb{R}^D$. According to the above, the assumption $f \sim \mathcal{GP}(0, \kappa(\boldsymbol{x}, \boldsymbol{x}'))$ becomes

$$\boldsymbol{\varpi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_D) \propto \exp\left(-\frac{1}{2}||\boldsymbol{\varpi}||_2^2\right),$$

where $\boldsymbol{I}_D \in \mathbb{R}^{D \times D}$ is an identity matrix. If we assume a Gaussian likelihood, we have

$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{X}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y_n|f(\boldsymbol{x}_n), \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(\langle \boldsymbol{\varpi}, \boldsymbol{\Phi}(\boldsymbol{x}_n)\rangle - y_n\right)^2\right).$$

Then, the posterior distribution is proportional to

$$\exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(\langle \boldsymbol{\varpi}, \Phi(\boldsymbol{x}_n)\rangle - y_n\right)^2 - \frac{1}{2}||\boldsymbol{\varpi}||_2^2\right).$$

We see that a maximum a posteriori–(MAP) estimation for the posterior will become a minimization problem with Tikhonov regularization [46], where the regularization parameter is related to the noise variance.

1.3.2 Learning from Multiple Annotators

A supervised learning scenario involves the estimation of a function $f : \mathcal{X} \to \mathcal{Y}$. Usually, each $\boldsymbol{x}_n$ is assigned to a single y_n, *i.e.*, the ground truth. Still, in several real-world problems instead of the ground truth, we have multiple labels provided by $R \in \mathbb{N}$ annotators with different levels of expertise [1]. It is common to find that each annotator r only labels $|N_r| \leq N$ samples, being $|N_r|$ the cardinality of the set $N_r \subseteq \{1, \ldots, N\}$ that contains the indexes of samples labeled by the r-th annotator. Besides, we define the set $R_n \subseteq \{1, \ldots, R\}$ holding the indexes of annotators that labeled the n-th instance. Thereby, it is possible to build a data set for the annotator $r \in \{1, 2, \ldots, R\}$, $\mathcal{D}_r = \{\boldsymbol{X}_r, \boldsymbol{y}_r\}$, where $\boldsymbol{X}_r = \{\boldsymbol{x}_n\}_{n \in N_r} \in \mathbb{R}^{|N_r| \times P}$ and $\boldsymbol{y}_r = \left\{y_n^{(r)}\right\}_{n \in N_r} \in \mathcal{Y}_r$ (commonly $\mathcal{Y}_r = \mathcal{Y}, \forall r$) are the input feature matrix and the labels given by the r-th annotator, respectively. Besides, $\boldsymbol{y}_r$ is composed of elements $y_n^{(r)}$, which is the r-th annotation of sample $\boldsymbol{x}_n$.
Now given the data set from multiple annotators $\mathcal{D} = \left\{\boldsymbol{X}, \boldsymbol{Y} = \{y_n^{(r)}\}_{n \in N_r \in R_n}\right\}$ the aims of a multi-labeler approach are: First, to estimate the unknown gold standard for the instances in the training set $\boldsymbol{y} = [y_1, \ldots, y_N]$. Second, to code the annotators behavior as a function of the input space. Finally, the third objective is to build a supervised learning model which generalizes well on unseen data [28]. A graphical comparison between a typically supervised learning dataset and a multi-labeler database is shown in Figure 1.2.

1.4 Literature Review on Supervised Learning from Multiple Annotators

As we established previously, *learning from crowds* faces supervised learning problems where the gold standard is not available. Among the developed works on this topic, we recognize two main groups: The approaches based on frequentist models (FM) and the Bayesian models (BM) schemes.

Regarding FM, we recognize the model proposed in [47]. Here, the authors first perform a geometrical interpretation of the majority voting (MV) scheme, and they remark that the estimated true label corresponds to the annotation (which can be seen as a point in a space generated by the MV approach) that has the largest distance to a decision hyperplane. Following this interpretation, they propose a weighted majority voting approach for binary and multi-class classification, where the weights are related to the annotator's reliability. Following the notion of margin in multi-class support vector machines (SVM), authors in [47] define the *crowdsourcing margin*, which is the minimal difference between the aggregated score of the potential true label

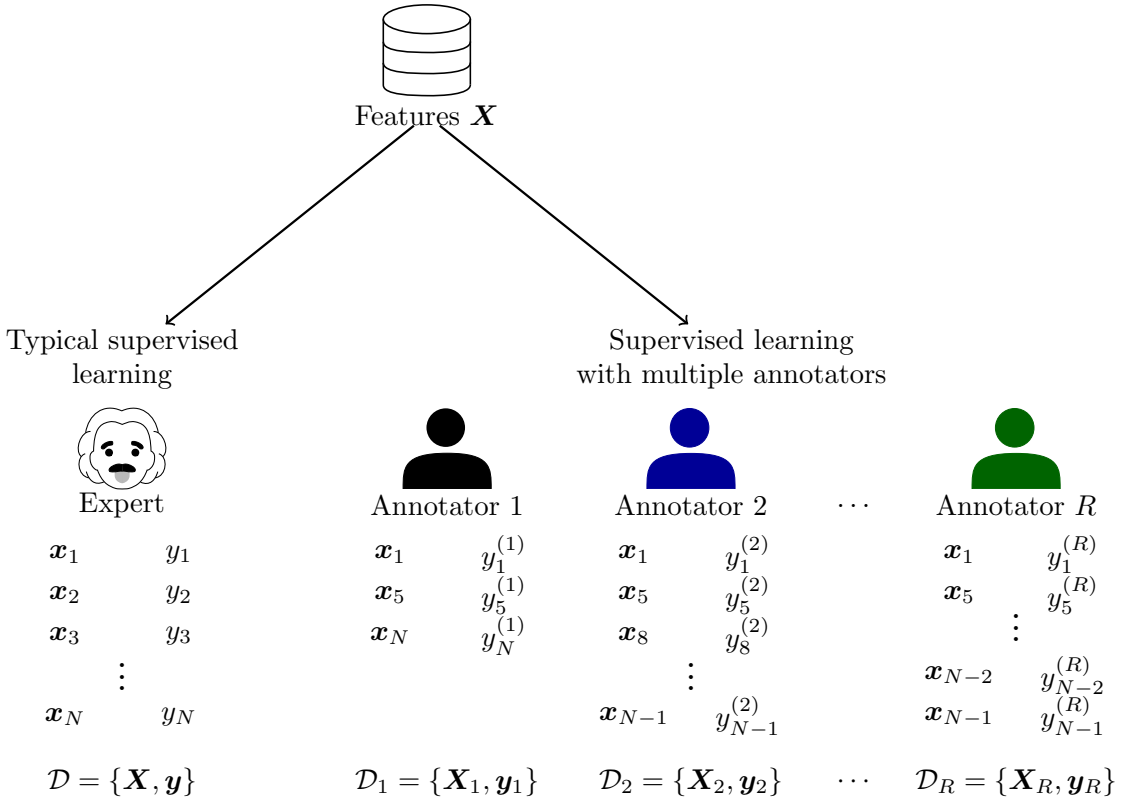

Figure 1.2 Graphical comparison between datasets for typical supervised learning settings and datasets from multiple annotators

and the scores for other alternative labels. Accordingly, the annotators' reliability is estimated as generating the largest margin between the potential true labels and other alternatives. However, this approach considers that the annotators' expertise is stationary across the input space; also, it does not consider account dependencies among the labelers. On the other hand, authors in [48] propose a mixture of classifiers aiming to deal with multi-labelers scenarios. The idea is to consider R classifiers $g_r(\boldsymbol{x})$, where each one is trained using the dataset for the r-th annotator $\mathcal{D}_r = \{\boldsymbol{X}_r, \boldsymbol{y}_r\}$, being $\boldsymbol{X}_r$ and $\boldsymbol{y}_r$ the inputs and their corresponding outputs given by labeler r, respectively. Accordingly, the proposed classifier is given by $f(\boldsymbol{x}) = \sum_{r=1}^{R} \nu_r g_r(\boldsymbol{x})$, where $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_R] \in \mathbb{R}^R$ are the weighting factors. Each parameter $\nu_r \in \mathbb{R}$ is estimated using a version of the variable ranking approach over a kernel matrix, which is computed based on the labels from the r-th annotator and the input features. Still, this approach lacks interpretability since it is unclear the meaning of the parameters $\boldsymbol{\nu}$. On the other hand, authors in [25] propose a Deep neural network–(DNN)-based approach to deal with multi-labeler data in both classification and regression tasks. The basic idea is to introduce an extra layer termed Crowdlayer, which allows training a DNN directly from the noisy labels from multiple annotators using backpropagation. The Crowdlayer is fed by what one usually defines as the output layer of a DNN (*e.g.* softmax/sigmoid for classification or linear for regression), and for each annotator r it learns a mapping from the output layer to the labels given by such labeler; hence, the Crowdlayer codes the annotators' reliabilities and biases.

So far, we have shown FM based on deterministic formulations such as SVM or DNN. Now, we focus on FM based on a probabilistic formulation. The most representative work is the one proposed in [1], which corresponds to an extension of the early work in [18] aiming to jointly learn a logistic regression-based classifier and the performance of the annotators in terms of sensitivity and specificity. Accordingly, the likelihood function (marginalizing for $\boldsymbol{y}$) is given by:

$$p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{a},\boldsymbol{b}) = \prod_{n=1}^{N}\prod_{r=1}^{R} p_n p\left(y_n^{(r)}|y_n=1,a_r\right) + (1-p_n)p\left(y_n^{(r)}|y_n=1,1-b_r\right), \quad (1.12)$$

where p_n is set as a logistic regression model with parameters $\boldsymbol{\omega} \in \mathbb{R}^P$

$$p_n = p(y_n=1|\boldsymbol{x}_n,\boldsymbol{\omega}) = \frac{1}{1+\exp(-\boldsymbol{\omega}^\top \boldsymbol{x}_n)}. \quad (1.13)$$

In addition, $p(y_n^{(r)}|y_n=1,a_r)$ and $p(y_n^{(r)}|y_n=0,1-b_r)$ are modeled as Bernoulli distributions with parameter $a_r \in [0,1]$ and $b_r \in [0,1]$ respectively, indicating the annotator's sensitivity and specificity. Figure 1.3 shows the graphical model for the approach in [1]. The model parameters $\boldsymbol{a}=[a_1,\ldots,a_R]^\top \in \mathbb{R}^R$, $\boldsymbol{b}=[b_1,\ldots,b_R] \in \mathbb{R}^R$, and $\boldsymbol{w}$ are estimated by maximizing the likelihood function in equation (1.12) and using the Expectation-Maximization–(EM) algorithm.

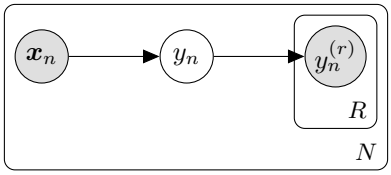

Figure 1.3 Graphical model for the approach proposed in [1]. $\boldsymbol{x}_n$, y_n, and $y_n^{(r)}$ are, respectively, the input, the ground truth, and the label given by the r-th annotator. Shaded nodes represent observed values, while unshaded nodes indicate latent variables

The model exposed below has significant relevance since they have inspired many works in the area of *learning from crowds*, including binary classification [5, 19, 20, 49], multiclass classification [16, 27], regression [22, 23], and sequence labeling [24].

According to the above, we notice some drawbacks. Firstly, such type of methods does not take into account the uncertainty in the predictions. Moreover, these approaches do not consider dependencies among the labelers nor the relationship between the annotators' expertise and the input space, which does not fit real-world scenarios as we established in the problem statement.

On the other hand, few works have been focused on BM. In contrast to FM-based approaches, BM allows modeling the uncertainty in the labels, making them robust in the presence of noise. We recognize three strategies that use BM to face supervised learning settings with multiple annotators based on the graphical model in Figure 1.3. The first method is the proposed in [19], termed Variational Gaussian processes for crowdsourcing–(VGPCR). The VGPCR's formulation

is based on the work given in [20]. Both use a GPs-based framework to solve binary classification problems and a sensitivity-specificity-based model to measure the annotators' behavior. Nonetheless, unlike [20], VGPCR treats the annotators' parameters (sensitivity and specificity) as random variables instead of fixed points. Also, it uses variational inference (instead of Expectation Propagation–(EP)) to estimate the posterior distributions of the model random variables [19]. In turn, [49] proposed a scalable version of VGPCR, named Scalable variational Gaussian processes for crowdsourcing–(SVGPCR). Authors in [49] argue that a classical GP has a computational complexity of $\mathcal{O}\left(N^3\right)$ caused by the inversion of kernel matrix $\boldsymbol{K_{ff}}$, which is prohibited for large datasets. In contrast, they propose a GP sparse approximation via the so-called Variational Fourier features–(VFF) within a variational inference framework to learn a new kernel matrix to approximate $\boldsymbol{K_{ff}}$ and to estimate the model's random variables (the same as in VGPCR). In such a way, the computational complexity is reduced to $\mathcal{O}(ND_f^2 + ND_fP)$, where $D_f \in \mathbb{Z}^+ << N$. Then, SVGPCR is extended by authors in [16] to deal with multiclass classification problems in the context of multiple annotators. In this model, the behavior of each annotator is assessed in terms of her/his confusion matrix, which is assigned to a Dirichlet prior to performing a fully Bayesian inference. Finally, regarding regression settings, authors in [22] propose a GP-based model, where the labels are assumed to be a corrupted version of the hidden true labels by Gaussian noise; thus, $y_n^{(r)} = y_n + \mathcal{N}(0, \sigma_r)$, where $\sigma_r \in \mathbb{R}^+$ is the r-th annotator error-variance.

Similarly to FM, most BM are based on two assumptions: the outputs of the labelers do not depend on the input features and independence among the annotators. These assumptions have been widely discussed in the problem statement, where we have established that they do not fit real-world scenarios. Next, we describe the FM and BM that try to relax these assumptions.

Approaches to Code the Relationship between the Annotators' Performance and the Input Space

Regarding the approaches that model the annotators' behavior as a function of the input samples, we recognize that they are based on FM with probabilistic formulation; in fact, they are based on the graphical model shown in Figure 1.4. The first work is proposed in [28, 50]. They

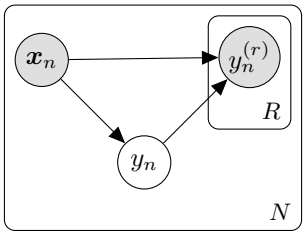

Figure 1.4 The graphical model used to code the labelers' expertise as a function of the input space. $\boldsymbol{x}_n$, y_n, and $y_n^{(r)}$ are the input, the ground truth, and the label given by the r-th annotator, respectively. Shaded nodes represent observed values, while unshaded nodes indicate latent variables

introduce an algorithm for binary classification, holding the joint conditional distribution as:

$$p(\boldsymbol{Y}, \boldsymbol{y}|\boldsymbol{X}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n) \prod_{r=1}^{R} p(y_n^{(r)}|\boldsymbol{x}_n, y_n), \tag{1.14}$$

where $p(y_n|\boldsymbol{x}_n)$ if fixed as a logistic regression model (see equation (1.13)). Similarly, they use two different models, for the term $p(y_n^{(r)}|\boldsymbol{x}_n, y_n)$:

$$p(y_n^{(r)}|\boldsymbol{x}_n, y_n) = \mathcal{N}\left(y_n^{(r)}|y_n, \eta_r(\boldsymbol{x}_n)\right), \tag{1.15}$$

$$p(y_n^{(r)}|\boldsymbol{x}_n, y_n) = (1 - \eta_r(\boldsymbol{x}_n))^{|y_n^{(r)} - y_n|} \eta_r(\boldsymbol{x}_n)^{1-|y_n^{(r)} - y_n|}, \tag{1.16}$$

where $\eta_r(\boldsymbol{x}_n)$ is fixed as a logistic regression with parameters $\boldsymbol{\gamma}_r \in \mathbb{R}^P$. Then, an EM algorithm is used to compute the model parameters.

On the other hand, the work in [51] proposes an extension of the work in [1], aiming to model the dependencies between the annotators and the input space. For doing so, they assume that the distribution of the input space $\mathcal{X}$ can be approximated by using a Gaussian mixture model with L components. Hence, the author hypothesizes that each annotator r has a particular performance for each mixture component. Accordingly, they define $a_l^{(r)} \in [0, 1]$ and $b_l^{(r)} \in [0, 1]$ respectively as the sensitivity and specificity of the annotator r in the l-th mixture component, with $l \in \{1, \ldots, L\}$. Therefore, the likelihood function for the incomplete data is given as:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{n=1}^{N} \prod_{l=1}^{L} \prod_{r=1}^{R} p_n p(y_n^{(r)}|y_n = 1, a_l^{(r)}) + (1 - p_n) p(y_n^{(r)}|y_n = 0, 1 - b_l^{(r)}), \tag{1.17}$$

where p_n is fixed to be a logistic regression model. Besides, $p(y_n^{(r)}|y_n = 1, a_l^{(r)})$ and $p(y_n^{(r)}|y_n = 0, 1 - b_l^{(r)})$ are modeled with Bernoulli distributions with parameters $a_l^{(r)}$ and $1 - b_l^{(r)}$ respectively. Similar to the previous approach, the model parameters are estimated by maximizing the likelihood and employing the EM algorithm.

So far, the works we have discussed are intended for binary classification. Conversely, for regression settings, we only identify the work proposed by [52], which establishes the following generative model

$$y_n = f(\boldsymbol{x}_n) + \epsilon_f, \tag{1.18}$$

$$y_n^{(r)} = g_r(y_n, \boldsymbol{x}_n) + \epsilon_g, \tag{1.19}$$

where $\epsilon_f, \epsilon_g \in \mathbb{R}^+$ are modeled as Gaussian noise. Note that $f : \mathcal{X} \to \mathcal{Y}$ and $g_r : \mathcal{X} \times \mathcal{Y} \to \mathcal{Y}_r$ characterize the regression function and the annotators model, respectively. Here, the authors use a GPs-based approach for the regression function f; thus:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{ff}}\right), \tag{1.20}$$

where the covariance matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ is computed by using a kernel that involves a squared exponential term, a linear term, and a constant bias (see [52] for more details). Similarly, each g_r is also modeled as a GP; in this sense, the conditional distribution can be expressed as:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \prod_{r=1}^{R} \mathcal{N}\left(\boldsymbol{y}_r|\boldsymbol{0}, \boldsymbol{S}_r\right), \tag{1.21}$$

where $\mathcal{Y}_r$ is connected with $\mathcal{X}$ and $\mathcal{Y}$ via a covariance matrix $\boldsymbol{S}_r \in \mathbb{R}^{N\times N}$, where the element $n, n^{'}$ is computed using a particular kernel $\kappa_{\boldsymbol{S}}(\{y_n, \boldsymbol{x}_n\}, \{y_{n'}, \boldsymbol{x}_{n'}\})$ [52]. In this work, the latent functions f and $\{g_r\}_{r=1}^{R}$ are estimated by following the Maximum a posteriori–(MAP) solution.

Approaches to Code Independence between Annotators

One of the most common assumptions in learning from crowds is that the annotators make their decisions independently. However, as we remark in previous sections, that assumption is hardly plausible. Regarding this, we only recognize the work in [10] based on BM. Here, the authors introduce a covariance measure among the annotators; hence, the annotator model is expressed by

$$p(\boldsymbol{Y}|\boldsymbol{y}, \boldsymbol{\varphi}, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \mathcal{N}\left(\boldsymbol{\Upsilon}_n|y_n\boldsymbol{1}_R + \boldsymbol{\varphi}, \boldsymbol{\Sigma}\right), \tag{1.22}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{R\times R}$ is the covariance matrix of the R annotators, $\boldsymbol{1}_R \in \mathbb{R}^R$ is an all-ones vector, and $\boldsymbol{\varphi} \in \mathbb{R}^{(R}$ codes the bias of the annotators. Besides, $\boldsymbol{\Upsilon}_n \in \mathbb{R}^R$ contains the labels provided for the n-th instance. Here, the model parameters are computed using a Bayesian framework based on Gibbs sampling.

Figure 1.5 shows relevant state-of-the-art works. To summarize, we raise the following observations:

- In general terms, we notice that most of the models for multiple annotators are based on FM to solve mainly binary classification tasks.
- We recognize only three relevant works (based on FM) to code the relationship between the input space and the annotators' behavior. Two are based on linear classifiers, two solve binary classification, and the remaining method employs a GP-based framework to solve a regression problem. Further, we observe that none of these three approaches considers labelers' interdependencies.
- Concerning the codification of annotators' dependencies, we only identify a single algorithm based on BM to face regression settings. However, such a method does not code the labelers' performance as a function of the input space.

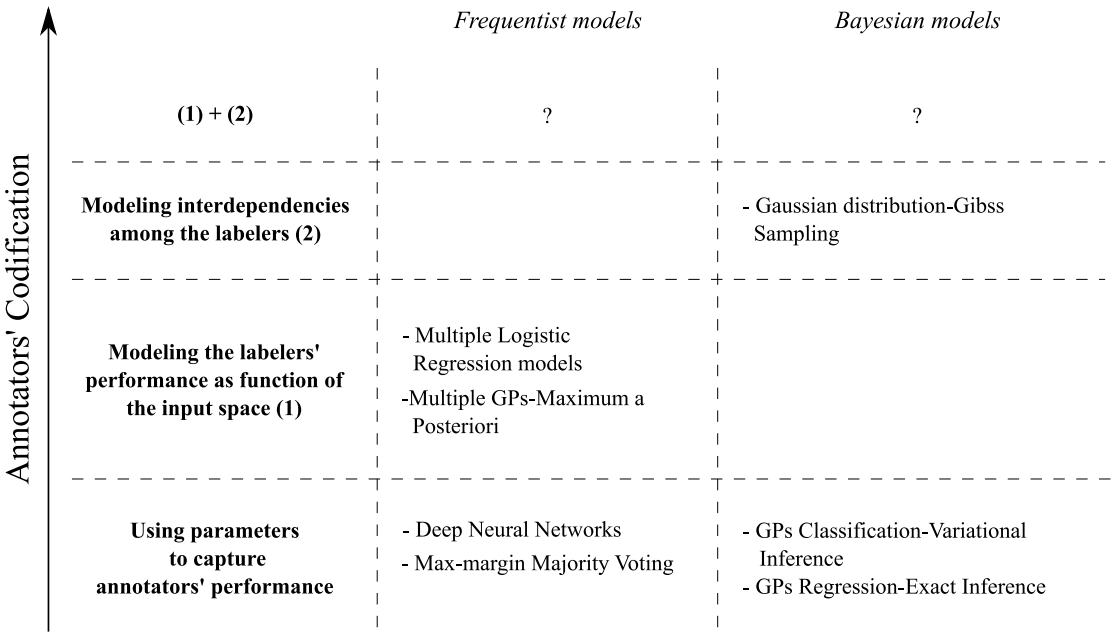

Figure 1.5 Relevant state-of-the-art works for supervised learning with multiple annotators

- Finally, we did not identify any model that codes the annotators' behavior under two assumptions: *i)* dependencies between the labelers; *ii)* the annotators' performance depends on the input space.

Accordingly, the principal aim of this research is to face the problems presented in Section 1.2 using both perspectives, Frequentist and Bayesian approaches.

1.5 Objectives

1.5.1 General Objective

To develop a supervised learning framework in the context of multiple annotators taking into account dependencies among the labelers and the fact that the annotators' performance is non-stationary across the input space, aiming to improve the representation of the labels in classification and regression tasks.

1.5.2 Specific Objectives

1. To develop a supervised learning approach (regression and classification) with multiple annotators, which uses a frequentist-based approach to code the labelers' expertise by considering dependencies among their decisions (labels).
2. To develop regression and classification approaches in multi-labelers scenarios, where the performance of the annotators is modeled using a frequentist-based strategy to code non-stationary and correlated labelers.
3. To develop a Bayesian-based method that jointly trains the supervised learning approach

(regression or classification) and estimates the labelers' performance by taking into account dependencies among them and the relationship between their performance and the input space.

1.6 Outline and Contributions

The main contributions are briefly introduced in the following sections, which are summarized in Figure 1.6.

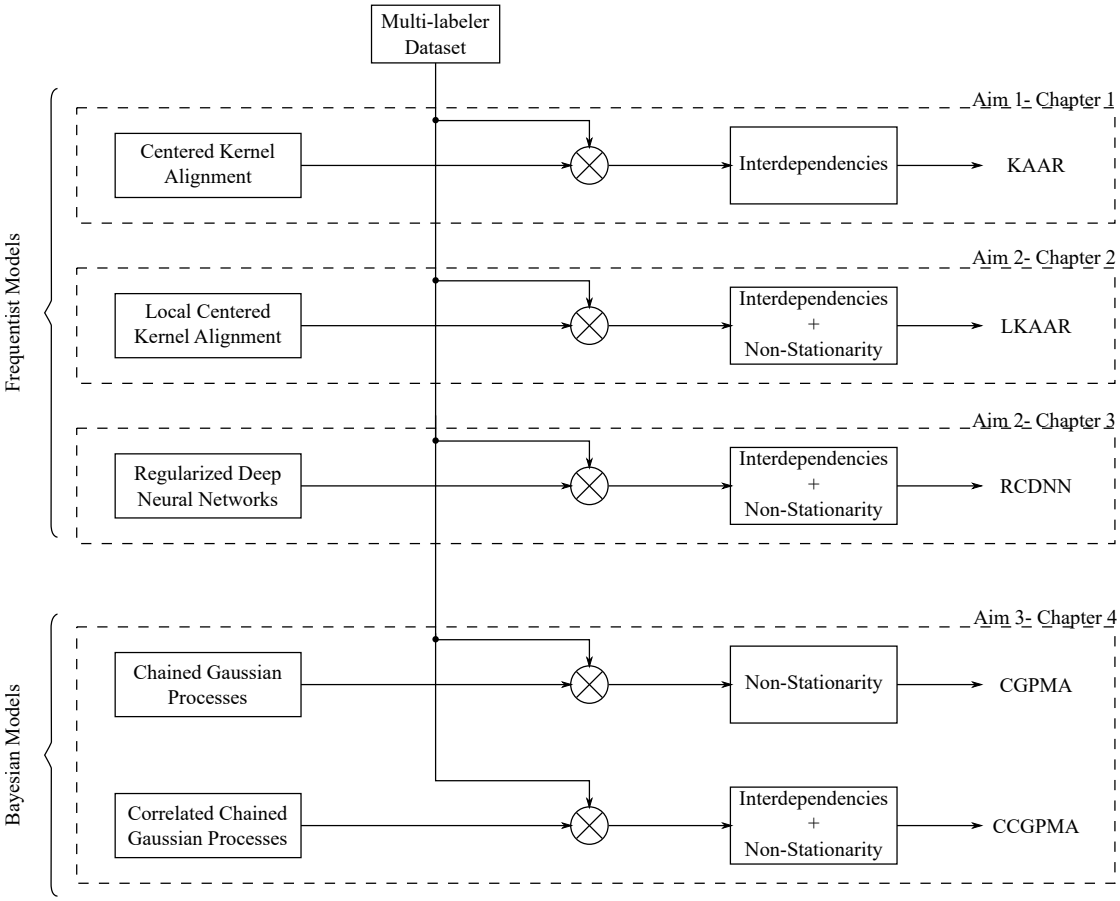

Figure 1.6 Relationship between the aims of this work and the developed methodologies. KAAR: Kernel alignment-based annotators relevance analysis. LKAAR: Localized kernel alignment-based annotator relevance analysis, RCDNN: Regularized chained deep neural network for Multiple Annotators, CGPMA: Chained Gaussian Processes for multiple annotators, and CCGPMA: Correlated chained Gaussian Processes for multiple annotators

1.6.1 Kernel Alignment-Based Annotator Relevance Analysis (KAAR)

The centered kernel alignment–(CKA) is commonly used for kernel selection in typical kernel-based learning models. The CKA approach comprises the computation of a convex combination of R kernels $\boldsymbol{K}_{\boldsymbol{\nu}} = \sum_{r=1}^{R} \nu_r \boldsymbol{K}_r$, where each matrix $\boldsymbol{K}_r$ is computed over the input features with a given kernel $\kappa_r(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$; also, the weights $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_R] \in [0,1]^R$ are computed by quantifying the similarities between the combination of kernels and the target kernel

$\boldsymbol{F}$, which in this case is a kernel computed over the labels (ground truth) $\kappa_{\boldsymbol{y}}(y_n, y_{n'})$ [53]. Our first contribution comprises the use of CKA to measure the labelers' performance. For doing so, we define a convex combination of R kernels, one for annotator; $\boldsymbol{K}_{\boldsymbol{\nu}} = \sum_{r=1}^{R} \nu_r \boldsymbol{K}_r$, where $\boldsymbol{K}_r$ is computed as $\kappa_r(y_n^{(r)}, y_{n'}^{(r)})$. Besides, we compute $\boldsymbol{\nu}$ as the average matching between the combination of kernels and the target kernel $\boldsymbol{F}$. We remark that for multi-labeler datasets, we do not have the actual labels; then, we choose the kernel over the features to be the target kernel $\kappa_X(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$. We hypothesize that the input features code the main patterns of the unknown gold standard labels.

We highlight that in the case of multiple annotators, each parameter ν_r is proportional to the r-th annotator's performance. In turn, a new sample label is predicted as a convex combination of learners adopting the achieved KAAR-based coding $f(\cdot) = \sum_{r=1}^{R} \nu_r g_r(\cdot)$, where each $g_r(\cdot)$ is a supervised learning algorithm trained with the dataset for the r-th algorithm $\mathcal{D}_r$ (Figure 1.7). Our approach estimates the performance of the annotators using a non-parametric model, allowing it to be more flexible concerning the labels' distribution. Moreover, our methodology relaxes the assumption of independence between the annotators, which codes possible correlations in the annotators' opinions to model their expertise. This approach is related to the first specific aim, and it is described in Chapter 2. Besides, such a methodology was published in [54].[1]

1.6.2 Localized Kernel Alignment-Based Annotator Relevance Analysis (LKAAR)

As we exposed in Section 1.6.1, the first contribution, KAAR, uses the CKA approach to model the annotators' interdependencies. CKA assumes that the distribution of the input features is stationary; that is, the weight ν_r penalizes equally all the samples for the r-th kernel function. Thereby, KAAR does not consider the relationship between the input features and the labelers' behavior, is unrealistic as established in Section 1.2. To deal with this issue, we introduce a localized multiple kernel learning-based approach to compute $\boldsymbol{K}_{\boldsymbol{q}} = \sum_{r=1}^{R} \boldsymbol{Q}_r \boldsymbol{K}_r \boldsymbol{Q}_r \in \mathbb{R}^{N \times N}$ [55, 56], where $\boldsymbol{Q}_r \in \mathbb{R}^{N \times N}$ is a diagonal matrix whose elements are defined by the vector $\boldsymbol{q}_r = [q_r(\boldsymbol{x}_1), \ldots, q_r(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$. The combination factors $\boldsymbol{q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_R]^\top \in \mathbb{R}^{N \times R}$ are estimated in such a way as to maximize the CKA between the kernel matrices $\boldsymbol{K}_{\boldsymbol{q}}$ and $\boldsymbol{F} \in \mathbb{R}^{N \times N}$, where we recall that for multiple annotators settings, $\boldsymbol{F}$ holds elements $\kappa_X(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$.

We remark that such combination factors $\boldsymbol{q}_r$ estimates the annotators' performance considering it as a function of the input features and considering the interdependencies among the labelers. Besides, like KAAR, LKAAR is built as a convex combination of classifiers, as shown in Figure 1.8; however, instead of having $\boldsymbol{\nu}$ as the combination coefficients, which are constant across the input space, we use $\boldsymbol{q}$. LKAAR has three remarkable features: i) the performance of each annotator is a function of the input space; ii) the assumption of independence among the

[1] A MATLAB implementation of KAAR is available in https://github.com/juliangilg/KAAR-Learning-from-multiple-annotators-using-Kernel-Alignment

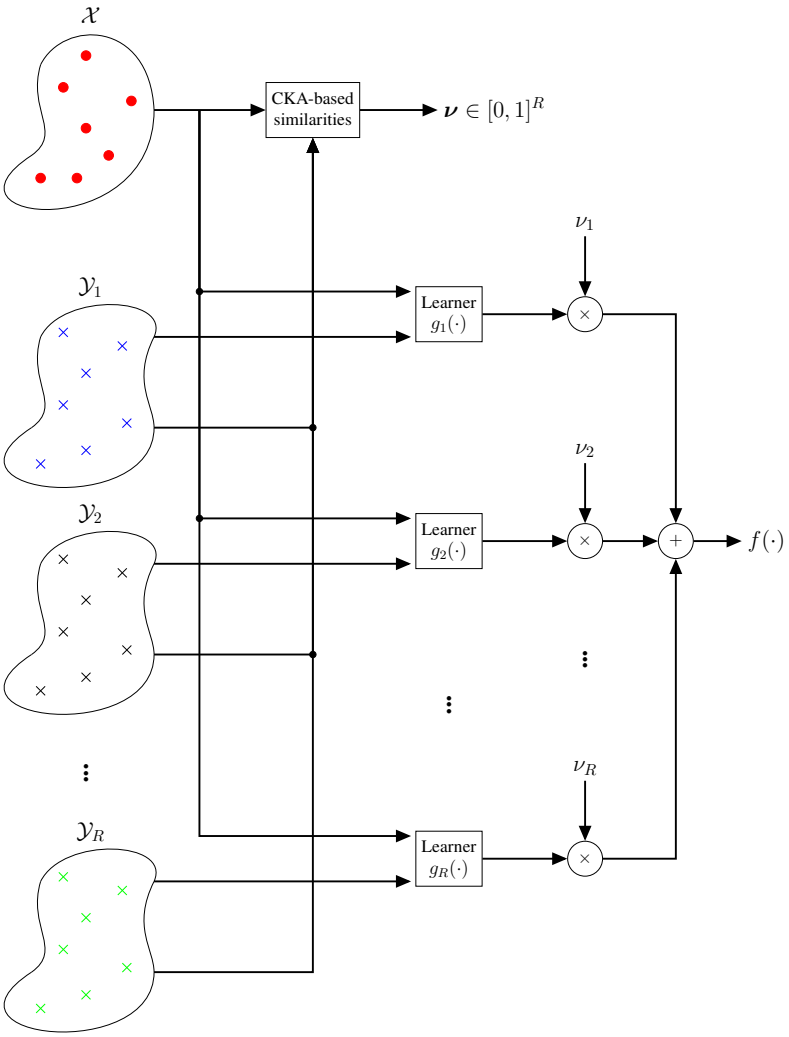

Figure 1.7 Kernel alignment-based annotator relevance analysis – (KAAR) pipeline. CKA stands for Centered Kernel Alignment. $\mathcal{X}$ indicates the input space, and each $\mathcal{Y}_r$ represents the output space for the r-th annotator. Moreover, $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_R]^\top$ is a vector containing the annotators' relevance parameters. Finally, the supervised learning mapping function f is computed as a convex combination of functions g_r that are trained from the r-th annotator's data

annotators is relaxed by modeling inter-annotators dependencies [57]; and iii) the performance of the annotators is estimated using a non-parametric model, allowing it to be more flexible to the distribution of the labels. This approach is related to the second specific aim, and it is described in Chapter 3, which is based on the publication [58].[2]

1.6.3 Regularized Chained Deep Neural Network for Multiple Annotators (RCDNN)

Our previous contributions, KAAR and LKAAR, deal with a multi-labeler problem in two stages; the first comprises the estimation of the annotator's parameters, which are then used

[2]A MATLAB implementation of LKAAR is available in https://github.com/juliangilg/LKAAR

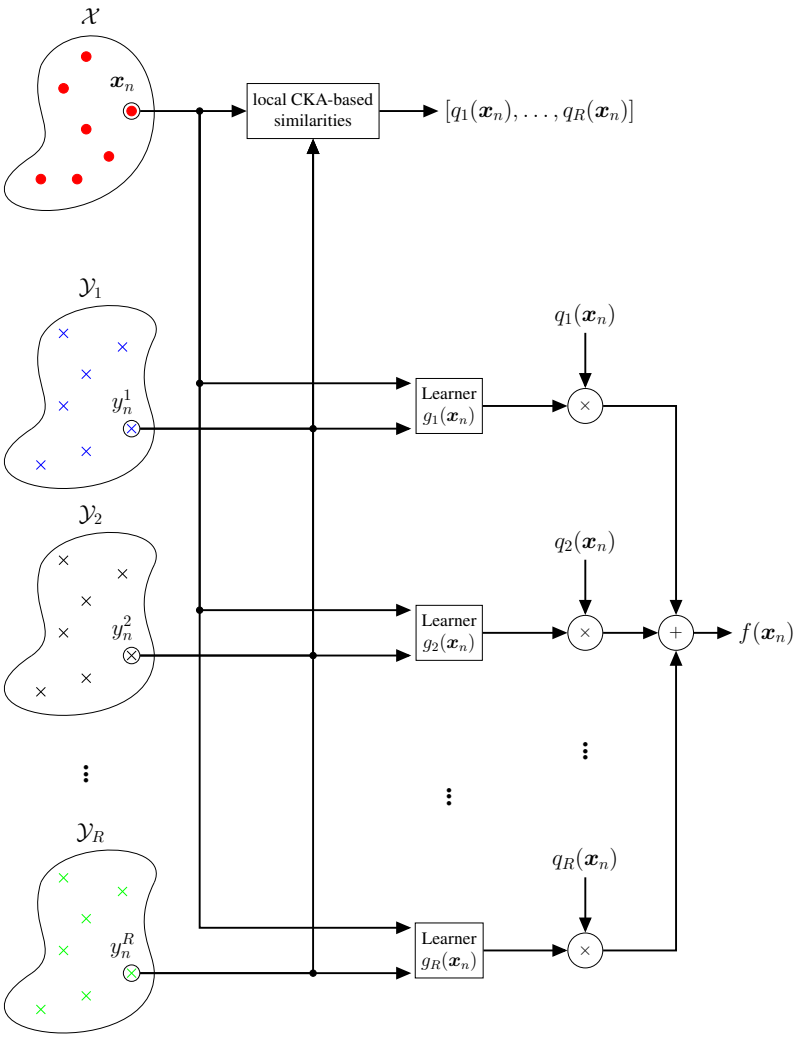

Figure 1.8 Localized kernel alignment-based annotator relevance analysis – (KAAR) pipeline. CKA stands for Centered Kernel Alignment. $\mathcal{X}$ indicates the input space, and each $\mathcal{Y}_r$ represents the output space for the r-th annotator. Moreover, $[q_1(\boldsymbol{x}_n), \ldots, q_R(\boldsymbol{x}_n)]^\top$ is a vector containing the annotators' relevance parameters for the n-th input sample. Finally, the supervised learning mapping function f is computed as a convex combination of functions g_r that are trained from the r-th annotator's data

as weight factors for the combination of R learning models (second stage). In contrast, our third contribution is a Regularized deep neural network-based method that jointly estimates the annotators' performance and the supervised learning algorithm. This approach is inspired by the chained Gaussian Processes model–(CGP) [59], where the idea is to model each parameter $\theta_j(\boldsymbol{x}),\ j \in \{1, \ldots, J\}$ in a given likelihood with multiple independent GPs priors (one GP prior per parameter). Unlike CGP, we consider that each neuron in the output layer of a DNN is linked to one of the parameters of a given likelihood through a deterministic function $h_j(\cdot)$. Thus, in a multi-labeler scenario, the annotators' parameters are modeled as a function of the input features. Moreover, we note that since each output in a DNN is computed as a linear combination of the outputs of a previous layer, our RCDNN codes interdependencies among the annotators. Besides, l1, l2, and Monte-Carlo Dropout-based regularizers are coupled within our method to deal with

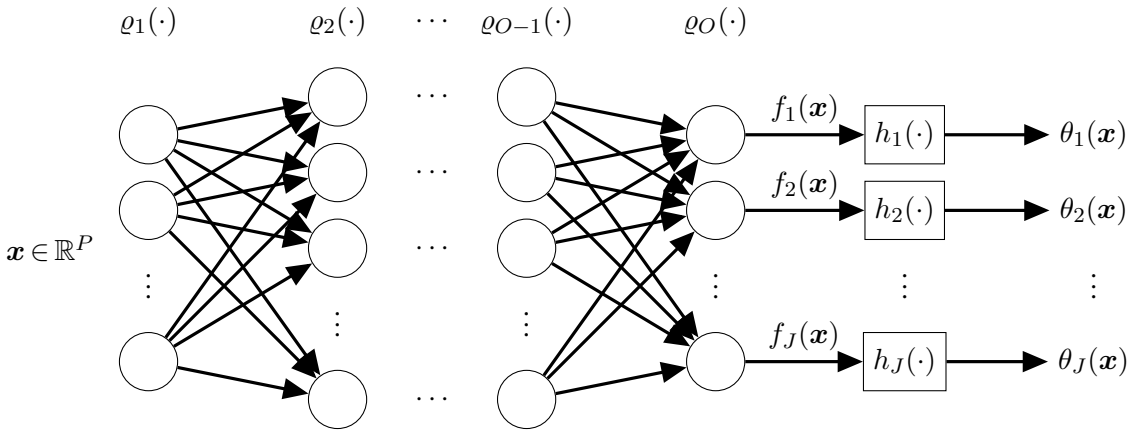

Figure 1.9 Regularized chained Deep Neural Network scheme. $\boldsymbol{x}$ is an input vector, $\varrho_o(\cdot)$ is the o-th layer of a DNN, $f_j(\boldsymbol{x})$ is the j-th output neuron for the input $\boldsymbol{x}$. Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood modeled as a function of the input space, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$

the overfitting issue in deep learning models. Like LKAAR, this approach is related to the second specific aim. RCDNN is described in Chapter 4 (based on [60]) and summarized in Figure 1.9.[3]

1.6.4 Chained Gaussian Processes for Multiple Annotators (CGPMA) and Correlated Chained Gaussian Processes for Multiple Annotators (CCGPMA)

Up to this point, our contributions, KAAR, LKAAR, and RCDNN, solve the multi-labeler problem from a frequentist perspective. Conversely, the last two contributions are based on Bayesian approaches, specifically on GPs. First, we apply the CGP [59] model to a multi-labeler likelihood. As exposed in Section 1.6.3, CGP links each likelihood parameter $\theta_j(\boldsymbol{x})$ to a GP prior $f_j(\boldsymbol{x}) \sim \mathcal{GP}(0, \kappa_{f_j}(\cdot,\cdot))$, being $\kappa_{f_j}(\cdot,\cdot)$ the covariance function and $j \in \{1,\ldots,J\}$. Such a connection between the parameter $\theta_j(\boldsymbol{x})$ and the $f(\boldsymbol{x})$ is performed via a deterministic function $h(\cdot) : \mathbb{R} \to \mathcal{M}_j$, where $\mathcal{M}_j$ is the domain for $\theta(\boldsymbol{x})$ (Figure 1.10). Accordingly, in a multi-labeler scenario, we are modeling the relationship between the annotators' performance and the input features.

Unlike CGP, we consider that multiple correlated GPs model the likelihood's parameters. For doing so, we take as a basis the ideas from a Multi-output GP–(MOGP) regression [40], where each output is coded as a weighted sum of shared latent functions via a semi-parametric latent factor model–(SLFM) [61]. In contrast to the MOGP, we do not have multiple outputs but multiple functions chained to the given likelihood parameters. Thus, each latent function $f_j(\boldsymbol{x})$ is computed as

$$f_j(\boldsymbol{x}) = \sum_{q=1}^{Q} w_{j,q}\mu_q(\boldsymbol{x}), \tag{1.23}$$

[3] A Python implementation of RCDNN is available in https://github.com/juliangilg/RCDNN-MA

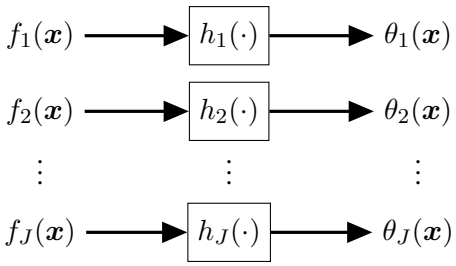

Figure 1.10 Chained Gaussian processes. Each $f_j(\boldsymbol{x})$ is a latent function that follows a GP, Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$

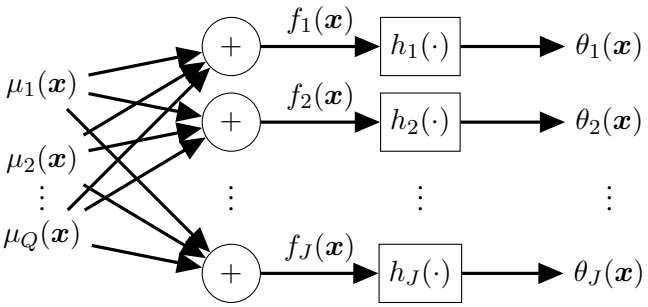

Figure 1.11 Correlated chained Gaussian processes. Each $f_j(\boldsymbol{x})$ is a latent function computed as a linear combination of functions $\{\mu_q(\boldsymbol{x})\}_{q=1}^{Q}$ that follows a GP. Moreover, each $\theta_j(\boldsymbol{x})$ is the j-th parameter of a given likelihood, and J is the number of likelihood parameters. Finally, $h_j(\cdot)$ is a deterministic function such that $\theta_j(\boldsymbol{x}) = h_j\left(f_j(\boldsymbol{x})\right)$

where $\mu_q(\cdot) \sim \mathcal{GP}(0, k_q(\cdot,\cdot))$, with $k_q : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is a kernel function, and $w_{j,q} \in \mathbb{R}$ is a combination coefficient ($Q \in \mathbb{N}$). Each, LF is chained to the likelihood parameters, as shown in Figure 1.11. From the multiple annotators' point of view, the likelihood parameters are related to the labelers' behavior; thereby, CCGPMA models the labelers' behavior as a function of the input features while also considering the annotators' interdependencies. CGPMA and CCGPMA are related to the third specific aim. Both models are described in Chapter 5, which is based on [62].[4]

1.6.5 Book Structure

This work is organized as follows. In Chapter 2, we introduce the KAAR model to code the annotators' performance taking into account dependencies among their labels, where the supervised learning algorithm is based on a linear combination of R learners. Chapter 3 discusses the LKAAR model, an extension of KAAR to model the relationship between the input features and the labelers' performance. In Chapter 4, we introduce a Regularized DNN-based approach to jointly train a supervised learning model and model the annotators' behavior considering their dependencies and the relationship between their performance and the input space. Chapter 5

[4] A Python implementation of CCGPMA is available in https://github.com/juliangilg/CCGPMA

presents the CGPMA and CCGPMA models, which address the multi-labeler problem from a Bayesian perspective. CGPMA considers the performance as a function of the inputs; while CCGPMA extends CGPMA to capture dependencies among the labelers. Finally, Chapter 6, describes the conclusions, future works, and academic discussion.

Chapter 2. Kernel Alignment-Based Annotator Relevance Analysis

Most works in the context of learning from multiple annotators are based on parametric models, which leads to the following issues: i) if the quantity of the parameters is comparable with the number of samples available for training, the model can fall quickly into overfitting [28]. ii) Linear or Gaussian constraints are commonly imposed to compute the optimal solutions analytically [63]. However, real-world datasets cannot fulfill such assumptions. iii) Most of the works assume independence between the annotators. However, it is suitable to consider that the labelers make their decisions independently. It is not true that these opinions are independent since there are possible correlations between the expert views [64].

In this chapter, we introduce a new kernel alignment-based annotator relevance analysis–(KAAR) approach to estimate the expertise of the labelers in scenarios where the gold standard is unavailable. KAAR computes each annotator's relevance as an averaged matching between the input features and the expert labels. A new sample label is predicted as a convex combination of learners adopting the achieved KAAR-based coding. Unlike previous works, our approach estimates the annotators' performance using a non-parametric model, allowing it to be more flexible concerning the distribution of the labels. Moreover, our methodology relaxes the assumption of independence between the annotators, highlighting possible correlations between the opinions to code their expertise. Our approach is similar to the proposed in [65] in that we perform the supervised learning task using a weighted combination of supervised learning algorithms. However, unlike such a work, we estimate the weights using a kernel alignment-based approach to quantify the matching between the input features and the annotator expertise. We empirically show, using both simulated and real annotators for regression and classification tasks, that our methodology can be used to estimate the performance of multiple labelers even if the gold standard is not available, outperforming state-of-the-art techniques.

2.1 Centered Kernel Alignment Fundamentals

In typical kernel-based learning models, selecting of a positive-definite function is crucial. Multiple kernel learning–(MKL) deals with such an issue by defining a kernel $\boldsymbol{K}_{\boldsymbol{\nu}}$ as a combination of R basis kernels; thus, we have [66]

$$\boldsymbol{K}_{\boldsymbol{\nu}} = h_{\boldsymbol{\nu}}\left(\{\boldsymbol{K}_r\}_{r=1}^{R} \,\middle|\, \boldsymbol{\nu}\right), \tag{2.1}$$

where $h_{\boldsymbol{\nu}}$ is the combination function, which can be linear (sum of kernels) or non-linear (*e.g.*,

product of kernels) [67]. Moreover, $\boldsymbol{K}_r \in \mathbb{R}^{N\times N}$ is computed using a particular kernel function over the input features $\boldsymbol{X}$, $\kappa_r : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and $\boldsymbol{\nu} \in \mathbb{R}^R$ is a vector containing the combination parameters. In particular, we define the function $h_{\boldsymbol{\nu}}$ as a convex combination; accordingly,

$$\boldsymbol{K}_{\boldsymbol{\nu}} = h_{\boldsymbol{\nu}}\left(\{\boldsymbol{K}_r\}_{r=1}^R \middle| \boldsymbol{\nu}\right) = \sum_{r=1}^{R} \nu_r \boldsymbol{K}_r, \tag{2.2}$$

where $\boldsymbol{\nu} \in [0,1]^R$, and $||\boldsymbol{\nu}||_2 = 1$ guarantees that $\boldsymbol{K}_{\boldsymbol{\nu}}$ is positive definite ($\boldsymbol{\nu} \in \mathbb{R}^R$ holds elements ν_r and $||\cdot||_2$ is the l2-norm). It is worth mentioning that adopting a convex sum favors the interpretability of parameters $\{\nu_r\}_{r=1}^R$, where the weight $\nu_r \in [0,1]$ represents the r-th kernel relevance [66].

Centered Kernel Alignment–(CKA)-based approaches leverage a data-driven estimator of $\boldsymbol{\nu}$ by quantifying the similarity among the kernels over input features and the kernel computed over the outputs $\boldsymbol{F} \in \mathbb{R}^{N\times N}$, which is also set as a positive definite function $\kappa_y : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$. Thereby, the matching between $\mathcal{Y}$ and $\mathcal{X}$ spaces can be measured through an empirical estimate of the CKA value $\rho \in [0,1]$ based on the $\boldsymbol{K}_{\boldsymbol{\nu}}$ and $\boldsymbol{F}$ kernels as follows [53]:

$$\rho\left(\boldsymbol{K}_{\boldsymbol{\nu}}, \boldsymbol{F}\right) = \frac{\langle \bar{\boldsymbol{K}}_{\boldsymbol{\nu}}, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}}}{||\,\bar{\boldsymbol{K}}_{\boldsymbol{\nu}}\,||_{\mathrm{F}}\,||\,\bar{\boldsymbol{F}}\,||_{\mathrm{F}}}, \tag{2.3}$$

where $\bar{\boldsymbol{F}}$ stands for a centered kernel matrix computed as: $\bar{\boldsymbol{F}} = \tilde{\boldsymbol{I}}\boldsymbol{F}\tilde{\boldsymbol{I}}$, being $\tilde{\boldsymbol{I}} = \boldsymbol{I} - \mathbf{1}^\top\mathbf{1}/N$ a centering matrix, $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ is the identity, and $\mathbf{1} \in \mathbb{R}^N$ is an all-ones vector. Namely, the vector $\boldsymbol{\nu}$ can be inferred by minimizing equation (2.3); the higher the ν_r value the better the contribution of $\boldsymbol{K}_r$ to match the target $\boldsymbol{F}$.

2.2 Kernel Alignment-Based Annotator Relevance Analysis

As we pointed out previously, CKA quantifies the similarity between a pair of input-output spaces to perform a proper configuration of a convex set of kernel functions. Alike, our idea is to use CKA to estimate the annotators' expertise in scenarios where the gold standard is unavailable. For doing so, we assume that the input features code the main patterns of the unknown gold standard labels. Thus, we measure the CKA between a kernel extracted from the input features (our target) and R kernels computed from the annotations given by each expert. Accordingly, we rewrite the matrix $\boldsymbol{F}$ as follows: $f_{nn'} = \kappa_X(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$, with $n, n' \in \{1, 2, \ldots, N\}$. For associating samples, several bivariate measures of similarity can be used, *e.g.*, linear, Gaussian, polynomial, etc. [68]. Here, to favor the mathematical tractability and to avoid the influence of the free parameters regarding the annotator performance, we fix κ_X as a linear kernel, yielding: $f_{nn'} = \boldsymbol{x}_n\boldsymbol{x}_{n'}^\top$. In turn, the pairwise similarity within the r-th expert annotations is defined by rewriting κ_r. The form κ_r will depend on the label space nature (*i.e.*, regression or classification), which is discussed in Section 2.2.1. On the other hand, the kernel matrix $\boldsymbol{K}_{\boldsymbol{\nu}}$, which codes the multiple annotation dependencies, is computed as a convex combination of the R kernels $\boldsymbol{K}_r$, holding

elements: $k_{nn'}^{(r)} = \kappa_r(y_n^{(r)}, y_{n'}^{(r)})$. Correspondingly, to capture each annotator performance, we compute the ν_r value using the following CKA-based optimization:

$$\hat{\boldsymbol{\nu}} = \arg\max_{\boldsymbol{\nu}} \rho(\boldsymbol{K}_{\boldsymbol{\nu}}, \boldsymbol{F}); \quad \text{s.t.} ||\boldsymbol{\nu}||_2 = 1. \tag{2.4}$$

Note that we highlight the dependence of the kernel matrix $\boldsymbol{K}_{\boldsymbol{\nu}}$ with respect to the vector $\boldsymbol{\nu}$. As a result, the weight $\hat{\nu}_r \in \hat{\boldsymbol{\nu}}$ in equation (2.4) explains the measured matching between the r-th expert, as stated in $\kappa_r(y_n^{(r)}, y_{n'}^{(r)})$, and the input features, as coded in $\kappa_X(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$. Moreover, through an auxiliary vector $\boldsymbol{v} \in \mathbb{R}^R$, wherein the following equality is imposed $\boldsymbol{\nu} = \boldsymbol{v}/||\boldsymbol{v}||_2$, the optimization problem in equation (2.4) can be solved trough the minimization of a quadratic cost function:

$$\hat{\boldsymbol{v}} = \arg\max_{\boldsymbol{v}} \mathcal{L}(\boldsymbol{v}) = \arg\max_{\boldsymbol{v}} \boldsymbol{v}^\top \boldsymbol{\Gamma} \boldsymbol{v} - 2\boldsymbol{v}^\top \boldsymbol{a}; \quad \text{s.t.} \quad ||\boldsymbol{v}||_2 \geq 0, \tag{2.5}$$

where the matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{R \times R}$ collects inter-annotator dependencies as:

$$\Gamma_{rr'} = \langle \bar{\boldsymbol{K}}_r, \bar{\boldsymbol{K}}_{r'} \rangle_{\mathrm{F}},$$

where $r, r' \in \{1, 2, \ldots, R\}$, and $\boldsymbol{a} \in \mathbb{R}^R$quantifies the similarity between the r-th expert and the input feature space as: $a_r = \langle \bar{\boldsymbol{K}}_r, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}}$. Accordingly, KAAR allows measuring the annotators' performance by taking into account dependencies among their behavior. A gradient descent-based approach is given to solve the optimization problem in equation (2.5) as follows:

$$\frac{d\mathcal{L}(\boldsymbol{v})}{\boldsymbol{v}} = 2\boldsymbol{\Gamma}\boldsymbol{v} - 2\boldsymbol{a}. \tag{2.6}$$

After estimation of the relevance vector $\boldsymbol{\nu}$, we assess the output $y_{new} \in \mathcal{Y}$ of a new input $\boldsymbol{x}_{new} \in \mathbb{R}^P$ as the following convex combination of learners

$$\hat{y}_{new} = \frac{1}{||\boldsymbol{\nu}||_1} \sum_{r=1}^{R} \nu_r g_r(\boldsymbol{x}_{new}), \tag{2.7}$$

where $g_r : \mathbb{R}^P \to \mathcal{Y}$ is a supervised learning algorithm trained from the set $\mathcal{D}_r$, and $||\cdot||_1$ is the l1-norm.

Notably, our kernel alignment-based annotator relevance analysis approach–(KAAR) can deal with missing labels. Also, it is possible to use any supervised learning scheme to learn the function g_r . To summarize, our KAAR (Algorithm 1.) counts on the enhancement of learning from multiple annotators by its two central stages: i) Seeking a relevance vector $\boldsymbol{\nu}$, relying on the averaged matching between the annotator labels and the input data features, and ii) Predicting the output of a new sample as a convex combination of learners adopting the achieved CKA-based relevance vector that intends to enhance the data separability based on the explained discrimination of each provided expert.

Algorithm 1: KAAR description

Data: $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, $\boldsymbol{x}_{new}$

1 Compute $\bar{\boldsymbol{K}}$ following equation (2.2).
2 Compute $\bar{\boldsymbol{F}}$ from the kernel function $\kappa_X(\cdot,\cdot)$.
3 Compute the annotators' parameters $\boldsymbol{\nu}$ by solving the optimization problem in equation (2.5).
4 Learn a function $g_r(\cdot)$ for each annotator by using $\mathcal{D}_r = \{\boldsymbol{X}_r, \boldsymbol{y}_r\}$.
5 Predict the output $\hat{y}_{new}$ for $\boldsymbol{x}_{new}$ following equation (2.7).

2.2.1 KAAR for Classification and Regression

One of the advantages of our proposal is that it can be applied for regression and classification. The key factor comprises the election of a proper function for $\kappa_r(y_n^{(r)}, y_{n'}^{(r)})$. We describe the kernel used for each setting (regression/classification).

Classification

In this first case, $y_n^{(r)} \in \{1, \ldots, K\}$, being K the number of classes. In turn, the pairwise similarity within the r-th expert annotations is defined by rewriting κ_r as follows:

$$\kappa_r(y_n^{(r)}, y_{n'}^{(r)}) = \begin{cases} 1, & \text{if } n, n' \in N_r \text{ and } y_n^{(r)} = y_{n'}^{(r)} \\ 0, & \text{Otherwise} \end{cases}, \tag{2.8}$$

where the condition $n, n' \in N_r$, inhibits the influence of missing labels.

Regression

For regression problems, we have the labels $y_n^{(r)} \in \mathbb{R}$. Thus, as in [53], we choose a linear kernel as follows

$$\kappa_r(y_n^{(r)}, y_{n'}^{(r)}) = \begin{cases} y_n^{(r)} y_{n'}^{(r)}, & \text{if } n, n' \in N_r \\ 0, & \text{Otherwise} \end{cases}, \tag{2.9}$$

where the condition $n, n' \in N_r$, inhibits the influence of missing labels.

2.3 Experimental Set-Up

In this section, we describe the experiments' configurations to validate our KAAR approach in multiple annotators scenarios for classification and regression tasks.

2.3.1 Classification

Testing Datasets

Aiming to test our KAAR approach as a classification tool from multiple annotators scenarios, we use eight datasets for binary classification of the well-known *UCI repository* [1]. The chosen datasets include: Wisconsin Breast Cancer Database–(breast), BUPA liver disorders–(bupa), Johns Hopkins University Ionosphere database–(ionosphere), Pima Indians Diabetes Database–(pima), Tic-Tac-Toe Endgame database–(tic-tac-toe), Iris Plants Database–(iris), Wine Data set–(Wine), and Image Segmentation Data Set–(Segmentation).

Moreover, the proposed approach is also tested on a real multiple annotators settings by applying it in two datasets. The first is a voice database, where the idea is to build a system for evaluating the voice quality. The Massachusetts Eye and Ear Infirmary Disordered Voice Database from the Kay Elemetrics company is used, which comprises voice records from healthy and different voice issues. Explicitly, a subset of $N = 218$ voice records is deemed: 51 healthy and 167 pathological. Four specialists assessed the quality following the GRBAS protocol, which comprises the evaluation of five qualitative scales: Grade of dysphonia–(G), Roughness–(R), Breathiness–(B), Aesthenia–(A), and Strain–(S). For each perceptual scale, the specialist assigns an integer tag ranging from 0 (healthy voice) to 3 (severe disease) [69]. Next, the well-known Mel-frequency cepstral coefficients–(MFCC) are computed for each voice signal to obtain an input space of 13 features ($P = 13$) [21]. The automatic assessment of voice quality configures five independent multi-class classification problems. However, since we have information about each voice record's diagnosis (pathological or normal), we map the labels to the set $\{1, 2\}$. The second dataset is called sentiment polarity, which corresponds to a collection of more than ten thousand sentences, is labeled as positive or negative. From this collection, $N = 5000$ sentences were selected randomly and published in the AMT platform to obtain labels from multiple annotators about each sentence's sentiment polarity. Besides, the remaining 5248 sentences were kept for testing. Each phrase is pre-processed to remove the stop words and is represented using a vocabulary with a size of 8919. A post-processing step based on Latent Semantic Analysis was carried out to reduce the dimensionality to $P = 1200$ features (for more details, [27]). Further, the music genre data is analyzed, holding a collection of song records labeled from one to ten depending on their music genre: classical, country, disco, hip-hop, jazz, rock, blues, reggae, pop, and metal. From this set, 700 samples were published randomly in the AMT platform to obtain labels from multiples sources (2946 labels were obtained from 44 workers). The feature extraction is performed by following the work by authors in [27] to obtain an input space with $P = 124$. Table 2.1 summarizes the tested datasets.

[1] http://archive.ics.uci.edu/ml

Table 2.1 Datasets description

	Name	Number of features	Number of instances	Number of classes
Semi-synthetic	Breast	9	683	2
	Bupa	6	345	2
	Ionosphere	34	351	2
	Pima	8	768	2
	Tic-tac-toe	9	958	2
	Iris	4	150	3
	Wine	13	178	3
	Segmentation	18	2310	7
Fully real	Voice	13	218	2
	Polarity	1200	10427	2
	Music	124	1000	10

Provided and Simulated Annotations

Regarding the UCI repository datasets, which are mainly used for typical supervised learning approaches without considering multiple annotators, two different simulation methods are studied to avoid biased results: *i) Classifier disrupting* proposed in [27], where a logistic regression classifier is trained using the input features and the true labels $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$, to obtain a classifier weight vector $\boldsymbol{w} \in \mathbb{R}^P$. Then, the label given by the r-th annotator is simulated as follows: $y_n^{(r)} = \boldsymbol{w}^\top \boldsymbol{x}_n + \mathcal{N}(0, \sigma_r^2)$, where $\mathcal{N}(0, \sigma_r^2)$ stands for a univariate normal distribution with zero mean and variance $\sigma_r^2 \in \mathbb{R}^+$. In turn, a sigmoid function is applied to map the disrupted label within the set $\{1, 2\}$. Accordingly, the higher the σ_r^2, value, the lower the expertise level of the r-th labeler. *ii) Biased coin* used in [50] that builds a binary number $\tau_n^{(r)} \in \{0, 1\}$ from a Bernoulli distribution ruled by the probability parameter $p_r \in [0, 1]$. Then, the simulated annotations of the r-th expert yields: $y_n^{(r)} = y_n$, if $\tau_n^{(r)} = 0$, otherwise, $y_n^{(r)} = \tilde{y}_n$, if $\tau_n^{(r)} = 1$; where $\tilde{y}_n$ is the flipped version of y_n.

Concerning the voice quality dataset, the annotations from four experts are provided, $R = 4$. For concrete testing, we only consider the R and B scales because, for scales A and S, the performance of the labelers is not satisfactory. For the scale G, the labelers' expertise is quite similar (according to the analysis performed in [21]). Similarly, we have labels from 203 workers for the polarity sentiment dataset. However, we only consider the annotators who labeled at least 15% of the available instances; in this sense, we use the information from $R = 7$ labelers. It is important to highlight that these 7 annotators do not label all the available instances. Further, for the music dataset, we only consider the annotators who labeled at least 20% of the instances; thus, we use the information from $R = 9$ labelers.

KAAR Training and Comparison Methods

KAAR employs a linear kernel to define the input data similarities and a closed form to compare expert annotations (equation (2.8)). It does not require any free parameter tunning to compute the relevance vector $\boldsymbol{\nu}$ in equations (2.4) and (2.5). Now, as the classification function

Table 2.2 A brief overview of the state-of-the-art methods tested

Algorithm	Description
GPC-GOLD	A GPC using the real labels (upper bound).
GPC-MV	A GPC using the majority voting of the labels as the ground truth.
MA-LFC [1]	A LRC with constant parameters across the input space.
MA-MAE [50]	A LRC where the sources parameters depend on the input space.
MA-DGRL [27]	A multi-labeler approach that considers as latent variables the annotator performance.
MA-GPC [20]	A multi-labeler GPC, which is as an extension of MA-LFC by using a non-linear approach.

Note: GPC: Gaussian Processes classifier, LRC: logistic regression classifier, MV: majority voting, MA: multiple annotators, MAE: Modelling annotators expertise, LFC: Learning from crowds, DGRL: Distinguishing good from random labelers.

g_r, we use a Gaussian processes-based model whose hyperparameters are set by optimizing a marginal likelihood [37]. Also, the validation is assessed by estimating the classification performance as the Area Under the Curve (AUC) for both simulated and real data. A cross-validation scheme is carried out with 30 repetitions where 70% of the samples are utilized for training and the remaining 30% for testing (except for the sentiment polarity dataset since it clearly defines the training and testing sets). On the other hand, Table 2.2 presents the comparison methods.

2.3.2 Regression

Testing Datasets

We test our approach using three types of datasets: fully synthetic data, semi-synthetic data, and fully real datasets. First, We generate *fully synthetic data* as a one-dimensional regression problem, where the ground truth for the n-th sample corresponds to $y_n = \sin(2\pi \boldsymbol{x}_n)$, where the input matrix $\boldsymbol{X}$ is formed by randomly sampling 300 points within the range $[0, 1]$ from a uniform distribution. The test instances are obtained by extracting 300 equally spaced samples from the interval $[0, 1]$. Second, to control the label generation [19], we build *semi-synthetic data* from six datasets related to regression tasks from the well-known UCI repository. We selected the following datasets: Auto MPG Data Set–(Auto), Bike Sharing Dataset Data Set–(Bike), Concrete Compressive Strength Data Set–(Concrete), The Boston Housing Dataset–(Housing),[2] Yacht Hydrodynamics Data Set–(Yacht), and Relative location of CT slices on axial axis Data Set–(CT). Third, we evaluate our proposal on one *fully real dataset*. In particular, we use the music genre data[3], holding a collection of songs record labeled from one to ten depending on their music genre: classical, country, disco, hip-hop, jazz, rock, blues, reggae, pop, and metal. From this set, 700 samples were published randomly in the AMT platform to obtain labels from multiples

[2]See https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html for housing

[3]http://fprodrigues.com/publications/learning-from-multiple-annotators-distinguishing-good-from-random-labelers/

Table 2.3 Datasets for regression

	Name	Number of features	Number of instances
Fully synthetic	synthetic	1	100
Semi-synthetic	Auto	8	398
	Bike	13	17389
	Concrete	9	1030
	Housing	13	506
	Yacht	6	308
	CT	384	53500
Fully real	Music	124	1000

sources (2946 annotations from 44 workers). Notice that the music dataset configures a 10-class classification problem; however, we are addressing a multi-class classification problem with a regression model in such an experiment. Such practice is not uncommon in machine learning and is usually known as "Least-square classification" [9, 37]. Table 2.3 summarizes the tested datasets for the regression case.

Simulated and Provided Labels

As we pointed out previously, *fully synthetic* and *semi-synthetic* datasets do not hold real annotations. Thus, it is necessary to generate these labels synthetically as a version of the gold standard corrupted by Gaussian noise, *i.e.*, $y_n^{(r)} = y_n + \epsilon_r$, where $\epsilon_r \sim \mathcal{N}(0, \sigma_r)$, being σ_r the r-th annotator error-variance for the sample n. This simulation method has been used in several works, such as [22, 23]. Further, for the music dataset, we only consider the annotators who labeled at least 20% of the instances; thus, we use the information from $R = 9$ labelers.

KAAR Training and Method Comparison

Alike in the classification case, KAAR for regression settings employs a linear kernel to define the input data similarities and a closed-form to compare expert annotations (see equation (2.8)), it does not require any free parameter tunning to compute the relevance vector $\boldsymbol{\nu}$ in equations (2.4) and (2.5). Besides, as the regression function g_r, we use a Gaussian processes-based model, whose hyperparameters are set by optimizing a marginal likelihood [37]. The quality assessment is carried out by estimating the regression performance as the coefficient of determination–(R^2). A cross-validation scheme is employed with 15 repetitions where 70% of the samples are utilized for training and the remaining 30% for testing (except for the *fully synthetic dataset* and music dataset since they clearly define the training and testing sets). Table 2.4 displays the methods employed by the state-of-the-art for comparison purposes. From Table 2.4, we highlight that for the model MA-DL, the authors provided three different annotators' codifica-

Table 2.4 A brief overview of state-of-the-art methods tested for regression tasks

Algorithm	Description
GPR-GOLD	A GPR using the real labels (upper bound).
GPR-Av MA-LFCR [1]	A GPR using the average of the labels as the ground truth. A LR model for MA where the labelers' parameters are supposed to be constant across the input space.
MA-GPR [20]	A multi-labeler GPR, which is as an extension of MA-LFCR. A Crowd Layer for DL, where the annotators' parameters are constant across the input space.
MA-DL [25]	

Note: GPR: Gaussian Processes Regression, LR: logistic regression, Av: average, MA: multiple annotators, DL: Deep learning, LFCR: Learning from crowds for regression.

tions: MA-DL-B, where the bias for the annotators is measured; MA-DL-S, where the labelers' scale is computed; and measured; MA-DL-B+S, which is a version with both [25].

2.4 Results and Discussion

2.4.1 Classification

First, a representative experiment is carried out to verify the KAAR capability to code the annotator performance. Namely, samples belonging to the Setosa and the Versicolour classes in the iris dataset are considered for testing the KAAR-based coding on a linearly separable problem. Afterward, we simulate five annotators using the simulation methods described in Section 2.3 by the parameters shown in Table 2.5. Moreover, Table 2.5 shows the KAAR-based coding results for the iris dataset regarding the matching weights in $\boldsymbol{\nu}$. We compute such weights as the CKA-based dependency between each expert kernel and the target for the two simulation methods. In particular, we analyze two different target kernels: from true labels and the input features ($\boldsymbol{\nu}_{TL}$ and $\boldsymbol{\nu}_{IF}$). Comparing the AUC for the simulated annotators and the KAAR-based results, overall, our approach can infer the annotators' performance from the input features. Figure 2.1 shows a visual comparison among the following kernels computed: the kernel calculated over the ground truth labels, the kernel estimated from the input features ($\boldsymbol{F}$), and the kernels estimated for each expert $\{\boldsymbol{K}_r : r = 1, \ldots, R\}$. Remarkably, the true labels and the input features kernels exhibit a significant coincidence; an appropriate parametrization reveals relevant information about the unknown gold standard. Besides, the annotation quality plays a vital role regarding the visual similarities between the true labels and the simulated annotations kernels; the lower the quality, the lower the visual similarity. The latter statement can be corroborated in Table 2.5, which shows the parameters used in each method to simulate annotators with different levels of expertise. The AUC is computed between the true labels and the simulated annotations. On the other hand, we recall that one of the main aims of this first proposal is to measure the annotators' performance by

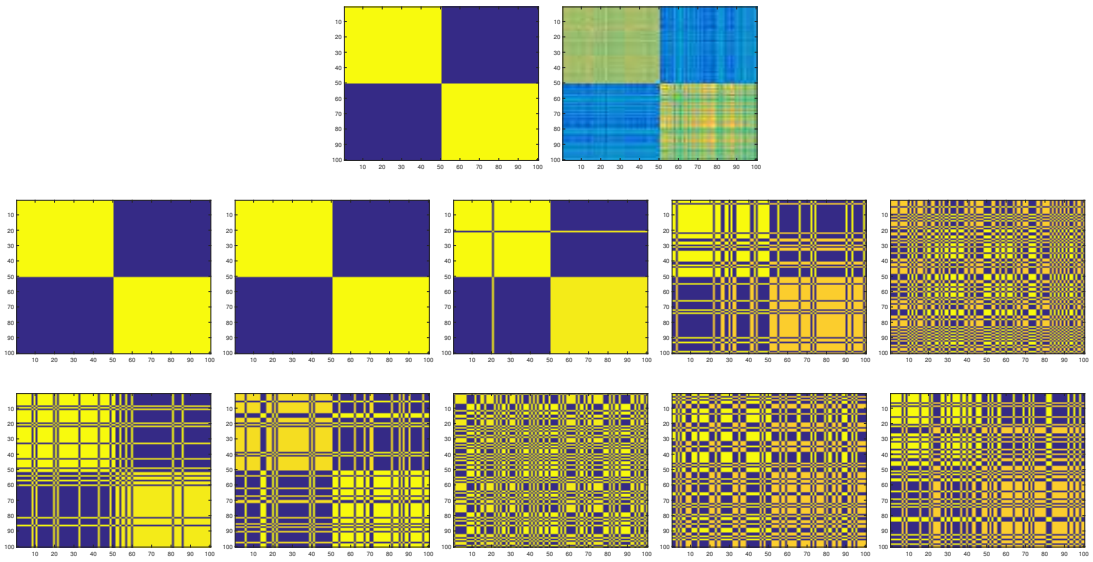

Figure 2.1 Iris dataset illustrative results. On the first row, from left to right, we show the similarities among samples according to the true labels and the input features. On the second and third rows, from left to right, we expose the kernels computed from the annotators' labels concerning each simulation method (see Table 2.5). Vertical and horizontal axes display the sample index, sorted by the value of the ground truth labels

Table 2.5 KAAR-based annotator coding results for the iris dataset

Simulation method	**Parameter value**	**AUC [%]**	$\boldsymbol{\nu}_{TL}$	$\boldsymbol{\nu}_{IF}$
Classifier disrupting	$\sigma_1^2 = 10$	100	0.7071	0.6971
	$\sigma_2^2 = 10^2$	100	0.7071	0.6971
	$\sigma_3^2 = 10^4$	99.88	0	0.1613
	$\sigma_4^2 = 10^6$	87.28	0	0.0465
	$\sigma_5^2 = 10^{10}$	45.72	0	0
Biased coin	$p_1 = 0.1$	87.00	0.8423	0.8410
	$p_2 = 0.3$	84.00	0.5116	0.5186
	$p_3 = 0.5$	51.00	0	0
	$p_4 = 0.6$	35.00	0.0767	0.0843
	$p_5 = 0.7$	31.00	0.1511	0.1291

Targets: the true labels ($\boldsymbol{\nu}_{TL}$) and the input features ($\boldsymbol{\nu}_{IF}$).

considering dependencies between them. Figure 2.2 presents a visual comparison between the five annotators' estimated dependencies. On the left, we show the Pearson correlation coefficient between the annotations $\boldsymbol{y}_r$ and $\boldsymbol{y}_{r'}$ for $r, r' \in \{1, \ldots, R\}$ as the reference value. We remark that only positive values of the Pearson correlation coefficient are allowed; negative values are fixed as 0. Similarly, on the right, we estimate the dependencies $\Gamma_{rr'} \in [0, 1]$ between the r-th and r'-th annotator based on the CKA formulation as $\Gamma_{rr'} = \langle \bar{\boldsymbol{K}}_r, \bar{\boldsymbol{K}}_{r'} \rangle_{\mathrm{F}}$. Remarkably, comparing the reference and the estimated dependencies among the annotators, we can see that, although the exact behavior is not recovered, KAAR can identify the labelers' relevant relationship, which improves the annotators' representation.

Up to this point, we have verified that the relevance vector $\boldsymbol{\nu}$ captures the performance of

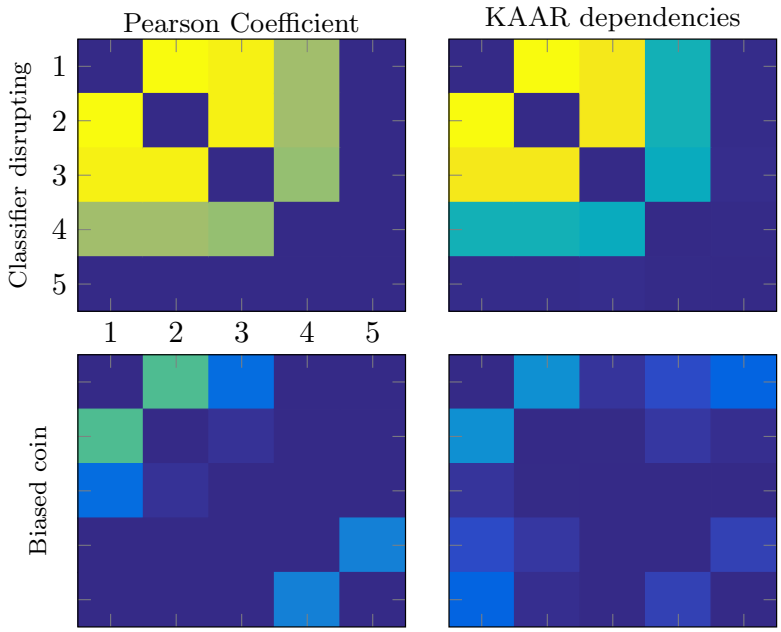

Figure 2.2 Annotators' interdependencies. In the left column, from top to bottom, we show the Pearson coefficient computed over the labels from multiple annotators. On the right, we present the annotators' dependencies estimated with our KAAR

multiple annotators even if the goal standard is unavailable and considers dependencies among the annotators. In turn, we perform classification experiments using datasets from the UCI repository. We simulate five annotators with different levels of expertise using the simulation parameters exposed in the second column of Table 2.5. Here, the KAAR-based coding is employed within a *Gaussian Process-based classifier*-(GPC) to predict a new label. For concrete testing, the GPC kernel function is fixed as a squared exponential with automatic relevance determination (ARD), and the kernel parameters are computed by minimizing the marginal likelihood function. Table 2.6 reports the mean and the standard deviation for the predicted AUC. Moreover, the method with the highest performance is highlighted in bold, excluding the upper bound (GPC-GOLD), which is a GPC trained with the true labels. Most of the classification methods from multiple annotators considered in this work outperform the majority voting baseline in most cases. This is not surprising since this baseline does not recognize different expertise. Remarkably, our approach is less prone to overfitting when compared with the parametric models (GPC-MV, MA-LFC, MA-MAE, MA-DGRL, MA-GPC). The above is a direct consequence of the number of parameters for modeling the performance of the annotators. For instance, MA-LFC and MA-GPC use $2 \times R$ parameters, MA-DGRL uses R parameters, and MA-MAE uses $P \times R$ parameters. Otherwise, our approach estimates the labelers' expertise as closed-form similarities between the input space and the labelers' annotations; hence, no free parameter tuning is required. However, it is important to highlight a trade-off between model overfitting and accuracy due to approaches that code the annotators' expertise using parametric representations, *i.e.*, MA-GPC and MA-LFC, which can capture biased labelers.

Table 2.6 UCI repository classification results

(a) Classifier disrupting.

Method	breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	average
GPC-GOLD	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	96.65 ± 3.81	99.40 ± 0.87	91.39 ± 2.05	92.18
GPC-MV	99.39 ± 0.41	65.15 ± 5.20	93.74 ± 3.04	82.59 ± 2.90	58.70 ± 5.11	$\mathbf{99.75 \pm 0.42}$	99.40 ± 0.87	94.13 ± 0.98	86.61
MA-LFC	99.51 ± 0.30	55.52 ± 17.25	86.97 ± 16.27	83.53 ± 1.73	62.89 ± 3.25	99.63 ± 0.39	$\mathbf{99.88 \pm 0.17}$	97.31 ± 0.42	85.66
MA-DGRL	99.61 ± 0.27	$\mathbf{68.11 \pm 7.62}$	90.51 ± 3.55	82.59 ± 2.82	61.10 ± 3.16	97.11 ± 1.87	99.78 ± 0.33	92.05 ± 2.36	86.36
MA-MAE	$\mathbf{99.62 \pm 0.31}$	66.37 ± 3.99	92.81 ± 2.83	83.10 ± 2.36	63.45 ± 2.80	99.55 ± 0.86	99.81 ± 0.27	95.59 ± 0.50	87.54
MA-GPC	95.08 ± 2.46	56.68 ± 10.39	94.79 ± 2.70	66.77 ± 4.30	60.07 ± 2.83	99.68 ± 0.34	99.81 ± 0.27	$\mathbf{98.20 \pm 0.22}$	83.88
KAAR	99.39 ± 0.39	65.33 ± 5.47	$\mathbf{96.63 \pm 1.57}$	$\mathbf{83.91 \pm 2.45}$	64.05 ± 3.58	99.64 ± 0.47	99.56 ± 0.81	94.50 ± 0.95	**87.87**

(b) Biased coin.

Method	breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	average
GPC-GOLD	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	96.65 ± 3.81	99.40 ± 0.87	91.39 ± 2.05	92.18
GPC-MV	94.13 ± 3.69	59.28 ± 5.95	65.48 ± 7.72	64.48 ± 8.57	62.54 ± 5.17	98.45 ± 1.84	97.83 ± 1.83	90.63 ± 2.73	79.10
MA-LFC	$\mathbf{99.35 \pm 0.43}$	$\mathbf{71.43 \pm 4.60}$	84.45 ± 4.27	$\mathbf{83.02 \pm 2.01}$	60.38 ± 3.09	$\mathbf{99.56 \pm 0.53}$	97.89 ± 1.78	99.41 ± 0.20	86.93
MA-DGRL	99.14 ± 1.26	68.89 ± 6.15	58.06 ± 8.45	82.23 ± 2.60	56.30 ± 4.63	94.01 ± 5.51	93.16 ± 5.77	96.24 ± 1.10	81.00
MA-MAE	96.57 ± 2.46	68.14 ± 4.58	67.34 ± 7.15	79.90 ± 3.49	50.43 ± 5.39	99.08 ± 0.92	95.25 ± 2.54	98.63 ± 0.33	81.92
MA-GPC	94.45 ± 2.51	43.19 ± 9.84	$\mathbf{92.03 \pm 6.05}$	68.96 ± 5.86	73.42 ± 4.15	99.45 ± 0.54	$\mathbf{99.68 \pm 0.35}$	$\mathbf{99.41 \pm 0.16}$	83.83
KAAR	99.04 ± 0.49	68.51 ± 5.42	85.51 ± 5.45	82.48 ± 2.72	$\mathbf{90.13 \pm 3.60}$	99.34 ± 0.91	99.33 ± 0.56	93.18 ± 1.78	**89.69**

Bold: the highest performance, excluding the upper bound (target) classifier GPC-GOLD.

The aforementioned is empirically demonstrated in Table 2.6; in some cases, MA-LFC and MA-GPC exhibit better performance than KAAR (mainly for the Bupa Dataset and segmentation). Nonetheless, our approach outperforms all the models considered for validation in general.

Table 2.7 outlines the achieved performances for the datasets holding real-world annotators. We highlight in bold the best method, excluding the upper bound (GP-GOLD). As seen for the voice quality dataset, the scale R allows exhibiting similar AUC values for most approaches. The latter can be explained in that, the annotators share comparable performances in terms of the AUC. Besides, we note that our KAAR approach obtains the highest AUC values for scale B, which is a remarkable result as long as, in this scale, the quality of the annotations decreases considerably for the R references. On the other hand, concerning the sentiment polarity dataset, the KAAR algorithm clearly outperforms all the validation methods; it achieves similar performance to the model trained with the gold standard (GPC-GOLD). The above can be explained in the sense that considering the correlations between the labelers encourages suitable coding about the annotators' performance, which positively affects on the quality of the predictions.

2.4.2 Regression

First, we perform a controlled experiment aiming to verify the capability of our KAAR method to estimate the performance of multiple annotators taking into account their dependencies. For this first experiment, we use the *fully synthetic* dataset described in Section 2.3.2. We simulate five labelers ($R = 5$) with different expertise levels using the parameters shown in Table 2.8. Besides, such a Table shows the KAAR-based coding results for this *fully synthetic* dataset regarding the matching weights in $\boldsymbol{\nu}$. From the classification result in Table 2.5, we note that our approach can measure the annotators' performance based on the similarities between the annotations from multiple labelers and the input features. Accordingly, for this first regression experiment, we compute the weights $\boldsymbol{\nu}$ as the CKA-based dependency between each expert ker-

nel and the target kernel, which is computed over the matrix $\boldsymbol{X}$. From Table 2.8, we remark that the behavior of R^2 score for the simulated annotators and the KAAR-based results are proportional; hence, we identify that our approach can infer the annotator performance from the input features in scenarios of real-valued labels.

Table 2.7 Real annotators datasets results

AUC([%])					
Method	**Voice Dataset**		**Polarity Dataset**	**Music Dataset**	**Average**
	R	**B**			
GPC-GOLD	92.83	91.89	80.26	92.84	89.45
GPC-MV	88.91	82.05	77.62	88.79	84.34
MA-LFC	90.86	86.68	50.53	85.99	78.51
MA-MAE	89.95	82.07	48.73	81.92	75.66
MA-DGRL	93.11	80.93	56.13	88.32	79.62
MA-GPC	**93.55**	80.16	61.18	82.53	79.35
KAAR (proposal)	92.37	**90.83**	**78.15**	**88.96**	**87.57**

Bold: the method with the highest performance, excluding the upper bound (target) classifier GPC-GOLD.

Now, Figure 2.3 (column 1) shows the regression results generated by 5 different regression schemes based on GPs, where the r-th regressor g_r is trained with dataset $\mathcal{D}_r$. Remarkably, if we compared the KAAR-based parameters $\boldsymbol{\nu}$ in Table 2.8 and the regression results. We note that the higher the regression performance, the higher the KAAR-based performance estimation. Moreover, in the first two images (from top to bottom) of the second column of Figure 2.3, we expose the result of our GPR-GOLD (a GP-based regression model trained with the actual labels) and the regression results generated by the KAAR-based approach based on the combination of the regression results exposed in the first column. We notice that our method performs the regression task properly. The regression results of KAAR are similar to the ones from GPR-GOLD.

On the other hand, we recall that our work's main objective is to estimate the annotators' performance by taking into account inter-dependencies. In the last two images of the second column in Figure 2.3, we expose a visual comparison between the five annotators' estimated dependencies. On the left, we show the Pearson correlation coefficient between the annotations $\boldsymbol{y}_r$ and $\boldsymbol{y}_{r'}$ for $r, r' \in \{1, \ldots, R\}$ as the reference value. We remark that only positive values of the Pearson correlation coefficient are allowed; negative values are fixed as 0. Similarly, on the right, we estimate the dependencies $\Gamma_{rr'} \in [0, 1]$ between the r-th and r'-th annotator based on the CKA formulation as $\Gamma_{rr'} = \langle \bar{\boldsymbol{K}}_r, \bar{\boldsymbol{K}}_{r'} \rangle_{\mathrm{F}}$. Comparing the reference and the estimated

Table 2.8 KAAR-based annotator coding results in the *fully synthetic* dataset

Parameter value	R^2	$\boldsymbol{\nu}$
$v_1 = 0.2$	0.5698	0.5632
$v_2 = 0.33$	0.2423	0.3048
$v_3 = 0.5$	0.0412	0.1252
$v_4 = 2$	-2.8083	0.0025
$v_5 = 4$	-8.0806	0.0040

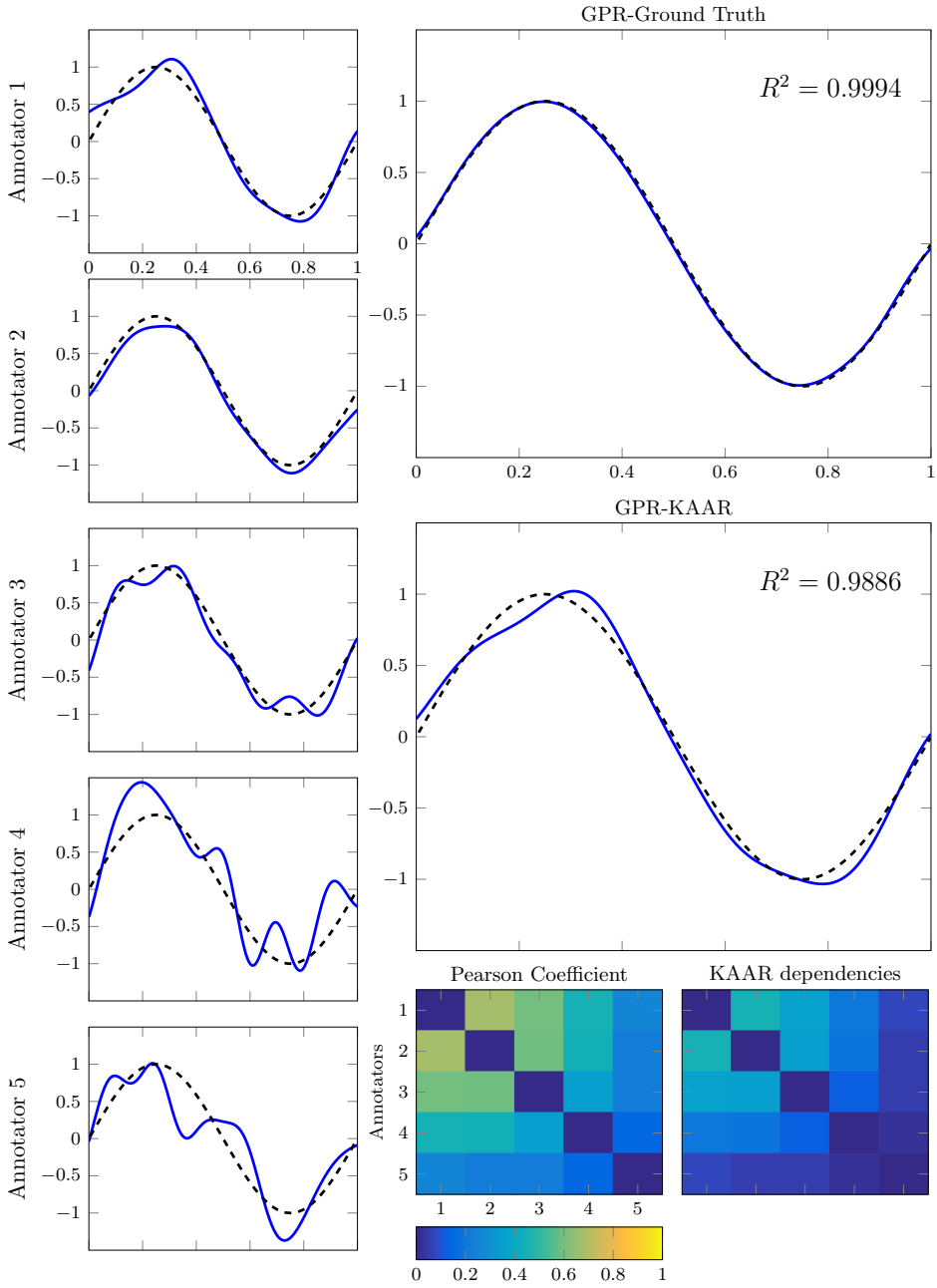

Figure 2.3 Regression illustrative results. In the first column, from top to bottom, we show the regression results for each annotator. In the second column, from top to bottom, we exhibit the result for GPR-GT, the result for our GPR-KAAR; finally, in the last row, we show the Pearson coefficient computed over the labels from multiple annotators and the dependencies estimated with our KAAR

dependencies among the annotators, we can see that, although the exact behavior is not recovered, KAAR can identify the labelers' relevant relationship.

On the other hand, regarding the semy-synthetic datasets, Table 2.9 shows the results of the *semi-synthetic datasets*. On average, our KAAR exhibits the best generalization performance regarding the R^2 score. Now, regarding its GPs-based competitors (GPR-Av, MA-GPR), we note that the intuitive lower bound GPR-Av exhibits a worse prediction than our KAAR. However, we remark on MA-GPR's behavior, which is lowest compared with its GPs-based competitors, even

Table 2.9 Regression results in terms of R^2 score over *semi-synthetic datasets*

Method	auto	Bike	concrete	housing	yacht	ct	Average
GPR-GOLD	0.8698 ± 0.0268	0.5612 ± 0.0059	0.8101 ± 0.0251	0.8288 ± 0.0389	0.8043 ± 0.0741	0.8652 ± 0.0054	0.7899
GPR-Av	0.8518 ± 0.0273	0.5432 ± 0.0076	0.7749 ± 0.0283	0.7984 ± 0.0424	0.7667 ± 0.0870	0.8223 ± 0.0074	0.7595
MA-LFCR	0.7992 ± 0.0231	0.3880 ± 0.0073	0.5975 ± 0.0388	0.7072 ± 0.0538	0.6120 ± 0.0775	0.6619 ± 0.2492	0.6276
MA-GPR	0.8558 ± 0.0269	0.4242 ± 0.0175	0.7807 ± 0.0312	0.7247 ± 0.0628	0.7319 ± 0.0965	0.0111 ± 0.0038	0.5881
MA-DL-B	0.7890 ± 0.0296	0.5926 ± 0.0121	0.2281 ± 0.0355	0.5323 ± 0.0757	0.1897 ± 0.0680	0.7590 ± 0.2437	0.5151
MA-DL-S	0.7866 ± 0.0274	0.5882 ± 0.0122	0.2259 ± 0.0312	0.5276 ± 0.0852	0.1900 ± 0.0722	0.8722 ± 0.2388	0.5318
MA-DL-B+S	0.7826 ± 0.0274	0.5874 ± 0.0135	0.2216 ± 0.0331	0.5326 ± 0.0884	0.1815 ± 0.1018	0.6782 ± 0.2659	0.4973
KAAR	$\mathbf{0.8541 \pm 0.0303}$	0.5667 ± 0.0062	$\mathbf{0.8034 \pm 0.0255}$	$\mathbf{0.8195 \pm 0.0354}$	$\mathbf{0.7729 \pm 0.0697}$	$\mathbf{0.8884 \pm 0.0022}$	**0.7841**

Note: Bold: the highest R^2 excluding the upper bound GPR-GOLD.

far worse than the supposed lower bound GPR-Av. The key to this particular outcome lies in its formulation; MA-GPR models the annotators' behavior by assuming no dependencies among the labelers' decisions.

Conversely, we analyze the results concerning the linear model MA-LFR; from the results, we notice that this approach's prediction capacity is far lower than our KAAR. Such an outcome indicates that most databases may have a non-linear structure. Finally, we analyze the DL-based models' results; we remark on a considerably low prediction capacity; in fact, they are even defeated by the linear model MA-LFR. We explain this behavior because the DL-based model uses an extra layer (termed CrowdLayer), which is used to manage the data from multiple annotators. Such a layer does not offer a suitable codification of the labelers' behavior.

Finally, the *fully real datasets* present the most challenging scenario, where both the input features and the labels come from real-world applications. Table 2.10 outlines the achieved performances. We notice that the DL approaches MA-DL-B+S and MA-DL-S obtain the best generalization performance in terms of the R^2 score, followed by our KAAR. Further, as theoretically expected, such performance lies between that of GPR-GOLD and GP-Av. Moreover, regarding the GPs-based competitor MA-GPR, we note that it exhibits the worst prediction capability with a R^2 close to zero. We argue that the above is a symptom of overfitting, which can be confirmed based on the training R^2 score is 0.4731, which is comparable with GPR-GOLD. Conversely,

Table 2.10 Regression results in terms of R^2 score over the *fully real dataset*

Method	Music
GPR-GOLD	0.4889
GPR-Av	0.2744
MA-LFCR	0.1404
MA-GPR	0.0090
MA-DL-B	0.2339
MA-DL-S	0.2934
MA-DL-B+S	**0.3519**
KAAR	0.2816

Note: Bold: the highest R^2 excluding the upper bound GPR-GOLD.

the linear approach MA-LFCR performs worse than the theoretical lower bound GP-Av, which indicates a non-linear structure in the Music dataset. Also, we observe that all regression models presented a lower generalization performance than previous results over the same dataset. The above is a repercussion of solving a multi-class classification problem with regression models.

2.5 Summary

We propose a new kernel-alignment-based approach, termed KAAR, to support the enhancement of supervised learning in the context of multiple annotators' data. To this end, KAAR computes each provided expert's relevance through a CKA-based averaged matching between the annotator labels and the input data features. A convex combination of supervised learning algorithms is carried out by adopting the multiple annotator performances coded in the KAAR-based relevance analysis. Unlike previous works, our proposal estimates the performance of the annotators using a non-parametric model. It relaxes the assumption of independence between the labelers, which allows the coding of some biases and tendencies between the annotators' opinions. We tested our approach in synthetic and real-world datasets for classification and regression settings. For the synthetic experiments, we use some databases from the UCI repository. We simulate multiple annotators following the schemes in Section 2.3.1 and Section 2.3.2 for classification and regression, respectively. On the other hand, we gather two classification tasks for the real-world datasets: the annotations are obtained from multiple experts' opinions (voice quality problem) and using the crowd-sourcing platform AMT (polarity and music data). Moreover, for regression problems, we use the music dataset. The results show that the proposed method can deal with binary classification, multiclass classification, and regression problems with multiple labelers. In fact, in most cases, our approach achieves competitive or even better results when compared to different state-of-the-art models [1, 20, 22, 25, 28, 70]. Furthermore, we experimentally demonstrate (using simulated annotators) that the performance of our approach does not depend on the model used for reproducing the annotations. Similarly, in the real-world datasets, it is evidenced that KAAR is not significantly affected when the labelers' expertise decreases drastically. Remarkably, KAAR deals with scenarios with missing labels.

Still, KAAR assumes that the annotators' performance only depends on the ground truth labels, which is not entirely accurate in many real applications, as it was pointed out in [28]. Accordingly, future work must be oriented toward relaxing this assumption by considering that the annotators' expertise depends on the ground truth and the instance they are labeling.

Chapter 3. Localized Kernel Alignment-Based Annotator Relevance Analysis

To model the annotators' behavior, learning parameters related to their performance is necessary. Such parameters include accuracy [27], the confusion matrix [21], error variance [22], and bias [23]. It is commonly found in the literature that the parameters are modeled as fixed points [20] or as random variables [16], where such parameters are homogeneous across the input data. The latter assumption is wrong since an expert makes decisions based not only on his/her expertise but also on the features observed from raw data [28, 50].

On the other hand, independence among the annotators is commonly used to reduce the complexity of the model [31] or based on the fact that it is plausible to guarantee that each labeler performs the annotation process individually [33]. Nevertheless, this is only partially correct because there may exist correlations among the annotators [51]. For example, if the sources are humans, the independence assumption is hardly feasible because knowledge is a social construction; hence, people's decisions will be correlated when they share information, communicate, or belong to a particular school of thought [34, 35]. Accordingly, the relaxation of this restriction could be used to improve the ground truth estimation [36].

According to the previously related, in this chapter, we propose an approach to face supervised learning problems with multiple annotators. Our model is an extension of the KAAR model introduced in Chapter 2. Like KAAR, our LKAAR is built as a convex combination of classifiers. The annotator's performance is computed by matching the input features and the labels. Similarly, both approaches take into account dependencies between the annotators. However, unlike KAAR, our LKAAR models the annotators' parameters as a function of the input features, which is an essential aspect of the behavior of the annotators, as has been established in the literature [27, 52]. Finally, we highlight that because our approach can model inconsistent annotators, it is more robust to outliers compared with models that do not consider the relationship between the input features and the labelers' behavior. LKAAR estimates the annotators' performance for every region in the input space; meanwhile, the other approaches estimate such performance as an average of some parameters [16, 19, 23]. Consequently, it is known that the average operator suffers under the presence of outliers [13].

3.1 Localized Kernel Alignment Fundamentals

As we exposed in Chapter 2, a usual approach to perform kernel selection comprises a convex combination of R basis kernels $\boldsymbol{K}_{\boldsymbol{\nu}} = \sum_{r=1}^{R} \nu_r \boldsymbol{K}_r$, where $\boldsymbol{K}_r \in \mathbb{R}^{N \times N}$ is a matrix holding elements $\kappa_r(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$, $\kappa_r : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ is a particular kernel ($n,\ n' \in \{1, \ldots, N\}$), and $\nu_r \in \mathbb{R}$

is a weighting factor.

The above combination assumes that the distribution of the input data is stationary. That is, the weight ν_r penalizes equally all the samples for the r-th kernel function, which cannot fulfill real-world scenarios. To deal with this issue, localized multiple kernel learning-based approaches compute $\boldsymbol{K_q}$ as the following quadratic combination [55, 56]:

$$\boldsymbol{K_q} = \sum_{r=1}^{R} \boldsymbol{Q}_r \boldsymbol{K}_r \boldsymbol{Q}_r, \tag{3.1}$$

where $\boldsymbol{Q}_r \in \mathbb{R}^{N\times N}$ is a diagonal matrix whose elements are defined by the vector $\boldsymbol{q}_r = [q_r(\boldsymbol{x}_1), \ldots, q_r(\boldsymbol{x}_N)]^\top$. The combination factors $\{\boldsymbol{q}_r\}_{r=1}^R$ should be estimated in such a way as to maximize the similarity between the kernel matrices $\boldsymbol{K_q}$ and $\boldsymbol{F} \in \mathbb{R}^{N\times N}$, where $\boldsymbol{F}$ holds elements $\kappa_y(y_n, y_{n'})$ and $\kappa_y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Then, the Centered Kernel Alignment–(CKA) between $\boldsymbol{K_q}$ and $\boldsymbol{F}$ is computed as follows [53]:

$$\rho\left(\boldsymbol{K_q}, \boldsymbol{F}\right) = \frac{\langle \bar{\boldsymbol{K}}_{\boldsymbol{q}}, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}}}{||\, \bar{\boldsymbol{K}}_{\boldsymbol{q}} \,||_{\mathrm{F}} \, ||\, \bar{\boldsymbol{F}} \,||_{\mathrm{F}}}, \tag{3.2}$$

where $\bar{\boldsymbol{F}}$ stands for a centered kernel computed as: $\bar{\boldsymbol{F}} = \boldsymbol{H}\boldsymbol{F}\boldsymbol{H}$, being $\boldsymbol{H} = \boldsymbol{I} - \mathbf{1}^\top \mathbf{1}/N$ a centering matrix, $\boldsymbol{I} \in \mathbb{R}^{N\times N}$ is the identity, $\mathbf{1} \in \mathbb{R}^N$ is an all-ones vector, and $||\cdot||_{\mathrm{F}}$ and $\langle \cdot, \cdot \rangle_{\mathrm{F}}$ stand for the Frobenius norm and inner product, respectively. Moreover, the centralized version of $\boldsymbol{K_q}$ is estimated as: $\bar{\boldsymbol{K}}_{\boldsymbol{q}} = \sum_{r=1}^{R} \boldsymbol{Q}_r \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r$, with $\bar{\boldsymbol{K}}_r = \boldsymbol{H}\boldsymbol{K}_r\boldsymbol{H}$.

3.2 Localized Kernel Alignment-Based Annotator Relevance Analysis

We enhance our KAAR presented in Chapter 2, where a CKA-based approach is applied to code each labeler's expertise in scenarios where the gold standard is not available. However, unlike KAAR, we consider the annotator's expertise a function of the input space. Accordingly, we assume that the input features in $\boldsymbol{X}$ contain the relevant information regarding the ground truth label. We compute the CKA between a kernel extracted over the input features and a local combination of R labelers kernels. Consequently, the target matrix $\boldsymbol{F}$ elements in equation (3.2) are computed through the kernel function: $\kappa_{\boldsymbol{X}} : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$. Further, to compute the local relevance of annotator r concerning the input $\boldsymbol{x}_n$, we infer the diagonal of $\boldsymbol{Q}_r$ in equation (3.1) as follows:

$$q_r(\boldsymbol{x}_n) = \begin{cases} \beta_0^{(r)} + \sum_{n'=1}^{N} \beta_{n'}^{(r)} \kappa_\beta(\boldsymbol{x}_n, \boldsymbol{x}_{n'}), & \text{if } n \in N_r \\ 0, & \text{Otherwise} \end{cases}. \tag{3.3}$$

The condition $n \in N_r$ inhibits the influence of missing labels, $\{\beta_j^{(r)} \in \mathbb{R}\}_{j=0}^N$ are combination coefficients, and $\kappa_\beta : \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ is a kernel function. Now, to compute $\beta_j^{(r)}$, we employ the

following CKA-based optimization problem:

$$\hat{\beta}_j^{(r)} = \arg\max_{\beta_j^{(r)}} \hat{\rho}\left(\boldsymbol{K_q}, \boldsymbol{F}\right) = \arg\max_{\beta_j^{(r)}} \frac{\langle \bar{\boldsymbol{K}}_{\boldsymbol{q}}, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}}}{|| \bar{\boldsymbol{K}} ||_{\mathrm{F}}}, \tag{3.4}$$

where:

$$|| \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}}^2 = \mathrm{tr}\left(\bar{\boldsymbol{K}}_{\boldsymbol{q}} \bar{\boldsymbol{K}}_{\boldsymbol{q}}^{\top}\right) = \mathrm{tr}\left(\sum_{r,l=1}^{R} \langle \boldsymbol{Q}_r \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r, \boldsymbol{Q}_l \bar{\boldsymbol{K}}_l \boldsymbol{Q}_l \rangle_{\mathrm{F}}\right) \tag{3.5}$$

codes the inter-annotator dependencies, being $\mathrm{tr}(\cdot)$ the trace operator. We have omitted the term $|| \bar{\boldsymbol{F}} ||_{\mathrm{F}}$ in equation (3.2) since it does not depend on the combination parameters $\beta_j^{(r)}$. A gradient descent-based approach is provided to solve the optimization problem in equation (3.4) as follows [71]:

$$\frac{\partial \hat{\rho}(\boldsymbol{K_q}, \boldsymbol{F})}{\partial \beta_j^{(r)}} = \mathrm{tr}\left(\left(\frac{\partial \hat{\rho}(\boldsymbol{K_q}, \boldsymbol{F})}{\partial \boldsymbol{Q}_r}\right)^{\top} \frac{\partial \boldsymbol{Q}_r}{\partial \beta_j^{(r)}}\right), \tag{3.6}$$

where $\frac{\partial \hat{\rho}(\boldsymbol{K_q}, \boldsymbol{F})}{\partial \boldsymbol{Q}_r} \in \mathbb{R}^{N \times N}$, yields:

$$\frac{\partial \hat{\rho}(\boldsymbol{K_q}, \boldsymbol{F})}{\partial \boldsymbol{Q}_r} = \frac{2 || \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}} \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r \bar{\boldsymbol{F}} - 2\langle \bar{\boldsymbol{K}}_{\boldsymbol{q}}, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}} || \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}}^{-1} \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r \bar{\boldsymbol{K}}_{\boldsymbol{q}}}{|| \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}}^2}, \tag{3.7}$$

$$\frac{\partial \hat{\rho}(\boldsymbol{K_q}, \boldsymbol{F})}{\partial \boldsymbol{Q}_r} = 2 \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r \left(\frac{\bar{\boldsymbol{F}}}{|| \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}}} - \frac{\langle \bar{\boldsymbol{K}}_{\boldsymbol{q}}, \bar{\boldsymbol{F}} \rangle_{\mathrm{F}} \bar{\boldsymbol{K}}_{\boldsymbol{q}}}{|| \bar{\boldsymbol{K}}_{\boldsymbol{q}} ||_{\mathrm{F}}^3}\right), \tag{3.8}$$

and $\frac{\partial \boldsymbol{Q}_r}{\partial \beta_j^{(r)}} \in \mathbb{R}^{N \times N}$ is a diagonal matrix holding elements:

$$\frac{\partial \left[\boldsymbol{Q}_r\right]_{nn}}{\partial \beta_j^{(r)}} = \begin{cases} 1, & \text{if } j = 0 \\ \kappa_{\beta}(\boldsymbol{x}_n, \boldsymbol{x}_j), & \text{if } j \neq 0 \text{ and } n \in N_r \\ 0, & \text{if } j \neq 0 \text{ and } n \notin N_r \end{cases}. \tag{3.9}$$

Next, to predict the output y_n, we propose the following convex combination of supervised learning models:

$$\hat{y}_n = \hat{g}(\boldsymbol{x}_n) = \frac{\sum_{r=1}^{R} q_r^2(\boldsymbol{x}_n) g_r(\boldsymbol{x}_n)}{\sum_{r=1}^{R} q_r^2(\boldsymbol{x}_n)}, \tag{3.10}$$

where $g_r : \mathbb{R}^P \rightarrow \mathbb{N}$ is a classification function learned from the r-th annotator's dataset $\mathcal{D}_r$. In turn, to compute each labeler's expertise $q_r(\boldsymbol{x}_{new})$ for a new sample $\boldsymbol{x}_{new}$, we hypothesize that an annotator exhibits comparable performance for similar instances:

$$q_r^2(\boldsymbol{x}_{new}) = \frac{\sum_{n \in N_r} q_r^2(\boldsymbol{x}_n) \exp\left(-\mathrm{d}\left(\boldsymbol{x}_{new}, \boldsymbol{x}_n\right)\right)}{\sum_{n \in N_r} \exp\left(-\mathrm{d}\left(\boldsymbol{x}_{new}, \boldsymbol{x}_n\right)\right)}, \tag{3.11}$$

where $\mathrm{d}(\cdot,\cdot): \mathbb{R}^P \times \mathbb{R}^P \to \mathbb{R}$ is the Euclidean distance. Finally, our localized kernel alignment-based annotator relevance analysis (LKAAR) can be summarized as in Algorithm 2.

Algorithm 2: LKAAR description

Data: $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, $\boldsymbol{x}_{new}$

1 Compute $\bar{\boldsymbol{K}}_{\boldsymbol{q}}$ following equation (3.1).
2 Compute $\bar{\boldsymbol{F}}$ from the kernel function $\kappa_X(\cdot,\cdot)$.
3 Solve the optimization problem: $\hat{\beta}_j^{(r)} = \arg\max_{\beta_j^{(r)}} \hat{\rho}\left(\boldsymbol{K}_{\boldsymbol{q}}, \boldsymbol{F}\right)$ in equation (3.4) from the gradient in equation (3.6).
4 Compute the annotators' parameters $q_r^2(\boldsymbol{x}_n)$, using equation (3.3) and $\hat{\beta}_j^{(r)}$.
5 Learn a classification function $g_r(\cdot)$ for each annotator by using $\mathcal{D}_r$.
6 Given a new sample $\boldsymbol{x}_{new}$ compute $q_r^2(\boldsymbol{x}_{new})$ using equation (3.11).
7 Predict the output $\hat{y}_{new}$ for $\boldsymbol{x}_{new}$ following equation (3.10).

3.2.1 LKAAR for Classification and Regression

Similar to KAAR, this second proposal has the advantage of being flexible to be applied to regression and classification scenarios. The key factor comprises the election of a proper function for $\kappa_r(y_n^{(r)}, y_{n'}^{(r)})$. For classification, we use the kernel function in equation (2.8); conversely, for regression we use the kernel function in equation (2.9).

3.3 Experimental Set-Up

This section describes the experiments' configurations to validate our LKAAR in multiple annotators scenarios for classification and regression tasks.

3.3.1 Classification

Testing Datasets

To test our LKAAR approach, we use three kinds of datasets. First, we generate *fully synthetic data* as two multivariate Gaussians in 2D ($P = 2$). The data for the first class is sampled from two multivariate Gaussian distributions $\mathcal{N}([-3,-3]^\top, \boldsymbol{\Sigma})$, and $\mathcal{N}([3,3]^\top, \boldsymbol{\Sigma})$; conversely, for the second class we use $\mathcal{N}([3,-3]^\top, \boldsymbol{\Sigma})$, and $\mathcal{N}([-3,3]^\top, \boldsymbol{\Sigma})$. The covariance matrix is fixed as: $\boldsymbol{\Sigma} = [0.8\ 0.1; 0.1\ 0.9]$.

Second, to control the label generation [19], we build *semi-synthetic data* from eight datasets devoted to binary and multi-class classification of the well-known UCI repository. Third, we evaluate our proposal on three *fully real datasets*, where both, the input features and the annotations

are captured from real-world problems. The description for these two dataset groups is found in Section 2.3.1; Table 2.1 summarizes the tested datasets.

Simulated and Provided Labels

As we pointed out, *fully synthetic* and *semi-synthetic* datasets do not hold real annotations. In this sense, generating these labels synthetically as a corrupted version of the gold standard is necessary. For this purpose, we use different simulation methods: *i) Biased coin* that assumes a constant annotator's performance across the input space; this method was already introduced in Section 2.3.1. *ii) Non-homogeneous labels*, which assume that the source performance depends on the input space [50]. In that sense, it represents the input space by R clusters (for concrete testing, we use the K-means algorithm to define each cluster). Then, the r-th annotator is fixed as the "expert," *i.e.*, his/her labels correspond to the ground truth in samples belonging to the cluster r. For the rest of the samples, the annotator makes mistakes in 35% of the cases, which are selected randomly. *iii) Biased coin (Non-homogeneous)* is an extension of a *Biased coin* [27]. It also divides the input space by R clusters. In each cluster, it samples a number $\tau_{n\sim c}^{(r)}$ ($n \sim k$ stands for the sample n belonging to the cluster $c \in \{1, 2, \ldots, R\}$) from a Bernoulli distribution with parameter $p_{c,r}$. Then, the simulated annotations of the r-th expert yields: $y_n^{(r)} = y_n$, if $\tau_{n\sim c}^{(r)} = 0$; otherwise, $y_n^{(r)} = \tilde{y}_n$, if $\tau_{n\sim c}^{(r)} = 1$. For the *fully real* data, Polarity and Music provides the crowd labels from the AMT platform. Regarding the voice quality dataset, we have annotations from four experts. However, for concrete testing, we only consider the G, R, and B scales, as suggested by authors in [21]. The annotators' expertise is not satisfactory for A and S scales.

LKAAR Training and Method Comparison

The Gaussian kernel is preferred in pattern classification because of its universal approximating ability and mathematical tractability [37]. Hence, we fix the κ_X and κ_β kernel functions in equations (3.3) and (3.4) as:

$$\kappa_X(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) = \kappa_\beta(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) = \exp\left(\frac{-||\,\boldsymbol{x}_n - \boldsymbol{x}_{n'}\,||_2^2}{2l^2}\right), \tag{3.12}$$

where $l \in \mathbb{R}^+$ is the bandwidth and $||\cdot||_2$ is the L2 norm. For concrete testing, we fix the term l as the median of the input distances [42]. Next, to model the classification function $g_r(\cdot)$ in equation (3.10), we use three different approaches: a multi-class linear classifier based on Logistic regression (LRC), a classifier based on support vector machines (SVMs) using a Gaussian kernel, and a Gaussian Processes classifier (GPC). The hyperparameters related to SVC are estimated by using a cross-validation scheme. Moreover, the GPC covariance function is computed via a squared exponential kernel, fixing the hyperparameters by optimizing the marginal likelihood [37]. The *one-vs-all* scheme is utilized in SVC and GPC to deal with multi-class problems.

The quality assessment estimates the classification performance as the Area Under the Curve

Table 3.1 A brief overview of the state-of-the-art methods tested

Algorithm	Description
GPC-GOLD	A GPC using the real labels (upper bound).
GPC-MV	A GPC using the majority voting of the labels as the ground truth.
MA-LFC [1]	A LRC with constant parameters across the input space.
MA-DGRL [27]	A multi-labeler approach that considers as latent variables the annotator performance.
MA-MAE [50]	A LRC where the source parameters depend on the input space.
MA-GPC [20]	A multi-labeler GPC, which is an extension of MA-LFC by using a non-linear approach.
KAAR [54]	A kernel-based approach that employs a convex combination of classifiers and codes labelers dependencies.

Note: GPC: Gaussian Processes classifier, LRC: logistic regression classifier, MV: majority voting, MA: multiple annotators, MAE: Modelling annotators expertise, LFC: Learning from crowds, DGRL: Distinguishing good from random labelers, KAAR: kernel alignment-based annotator relevance analysis.

(AUC) and the overall accuracy (Acc). Further, the AUC is extended for multi-class settings, as the authors in [72] discussed. A cross-validation scheme is employed with 30 repetitions, where 70% of the samples are utilized for training and the remaining 30% for testing (except for the music dataset since it clearly defines the training and testing sets). Table 3.1 displays the methods employed by the state-of-the-art for comparison purposes. The Matlab codes for our LKAAR and the state-of-the-art methods studied are publicly available.[1] We remark that the GPC-Gold only provides an upper bound for our LKAAR.

3.3.2 Regression

Testing Datasets

Aiming to test our LKAAR in regression scenarios, we use the synthetic, semi-synthetic, and real datasets presented in Section 2.3.2.

Simulated and Provided Labels

We recall that *fully synthetic* and *semi-synthetic* datasets do not hold real annotations from multiple labelers. In this sense, it is necessary to generate such labels synthetically as corrupted versions of the true labels, considering that the labelers' performance is a function of the input samples. Accordingly, we assume that the input space is represented by R clusters, where such representation is carried out from the K-means algorithm. Then, in cluster $c \in \{1, \ldots, R\}$ the simulated labels follow, $y_n^{(r)} = y_n + \mathcal{N}(0, \sigma_{c,r})$, where $\sigma_{c,r}$ represents the error variance for the r-th labeler in cluster c. For real datasets, we perform a similar procedure as in Section 2.3.2.

[1]GPC-MV, MA-LFC, MA-MAE, MA-DGRL, GPC-GTIC, KAAR, and LKAAR codes: https://github.com/juliangilg. MA-GPC codes: http://www.fprodrigues.com/

LKAAR Training and Method Comparison

Like in the classification problem, our LKAAR is built based on Gaussian kernels; in that sense, we define $\kappa_{\boldsymbol{X}}$ and κ_{β} as in equation (3.12). Next, we use a Gaussian processes-based model to estimate each function g_r, whose hyperparameters are estimated by minimizing a marginal likelihood [37]. The regression performance is measured using the coefficient of determination–(R^2). A cross-validation scheme is employed with 30 repetitions where 70% of the samples are utilized for training and the remaining 30% for testing (except for the music dataset, since it defines the training and testing sets). Table 3.2 displays the employed methods of the state-of-the-art for comparison; we remark that for the model MA-DL, the authors provided three different annotators' codification: MA-DL-B, where the bias for the annotators is measured; MA-DL-S, where the labelers' scale is computed; and measured; MA-DL-B+S, which is a version with both [25].

Table 3.2 A brief overview of state-of-the-art methods tested for regression tasks

Algorithm	Description
GPR-GOLD	A GPR using the real labels (upper bound).
GPR-Av	A GPR using the average of the labels as the ground truth.
MA-LFCR [1]	A LR model for MA where the labelers' parameters are supposed to be constant across the input space.
MA-GPR [20]	A multi-labeler GPR, which is as an extension of MA-LFCR.
MA-DL [25]	A Crowd Layer for DL, where the annotators' parameters are constant across the input space.
KAAR [54]	A kernel-based approach that uses a convex combination of regression approaches and codes the labelers dependencies

Note: GPR: Gaussian Processes Regression, LR: logistic regression, Av: average, MA: multiple annotators, DL: Deep learning, LFCR: Learning from crowds for regression.

3.4 Results and Discussion

3.4.1 Classification

We perform a controlled experiment aiming to verify the LKAAR capability to estimate the performance of inconsistent annotators as a function of the input space and take into account their dependencies. For this first experiment, we use the *fully synthetic* dataset described in Section 3.3.1. We simulate five labelers ($R = 5$) with different levels of expertise. For the *Biased coin* we fix $\boldsymbol{p} = [0.1,\ 0.3,\ 0.5,\ 0.6,\ 0.7]$, where the r-th component codes the corresponding labeler's performance. For the *Biased coin (Non-homogeneous)* approach, we divide the input space into five regions and define the following performance matrix $\boldsymbol{P} \in [0, 1]^{R \times R}$:

$$\boldsymbol{P} = \begin{bmatrix} 0.00 & 0.90 & 0.50 & 0.15 & 0.60 \\ 0.90 & 0.00 & 0.30 & 0.40 & 0.75 \\ 0.50 & 0.30 & 0.00 & 0.60 & 0.30 \\ 0.15 & 0.40 & 0.60 & 0.00 & 0.80 \\ 0.60 & 0.75 & 0.30 & 0.80 & 0.00 \end{bmatrix}, \tag{3.13}$$

holding elements $P_{c,r}$, which are related to the expertise of the r-th annotator in the region c. Accordingly, we note that the r-th annotator is an expert (its labels correspond to the ground truth) in the region $k = r$.

Figure 3.1 (column 1) shows the LKAAR-based matching weights $q_r^2(\boldsymbol{x}_n)$. These weights are computed as the CKA-based similarities between each kernel expert and the target kernel from the input features. Ideally, the target kernel would be calculated over the true labels; however, we are dealing with scenarios where these correct labels are not available. Comparing the parameters used for the simulation method *Biased coin (Non-homogeneous)* (see equation (3.13)) and the LKAAR-based results, we can note that our approach can infer the labeler performance. Thus, an appropriate parametrization allows for capturing relevant information about the hidden ground truth. Figure 3.1 (column 2) shows the decision boundaries generated by different Gaussian process classifiers, where each was trained with the labels from one of the annotators.

Also, we can elucidate that the classification results generated by each of the simulated annotators ($g_r(\cdot)$) vary depending on the input space. Remarkably, we also note that the higher the classifier discrimination, the higher the LKAAR-based performance estimation. Moreover, in the first two images (from top to bottom) of the third column of Figure 3.1, we expose the result of our GPC-GOLD (a Gaussian processes classifier trained with the actual labels) and the decision boundaries generated by our LKAAR-based approach as in equation (3.10). We can observe that our method performs the classification task properly. Its decision boundaries are similar to those produced by the GPC-GOLD. As seen, the $q(\boldsymbol{x}_n)^{(r)}$ weights enhance the decision boundaries around regions where the r-th expert exhibits high performances. Besides, one of the main aims of our work is to estimate the annotators' performance by taking into account inter-dependencies. So, in the last two images of the third column in Figure 3.1, we expose a visual comparison between the estimated dependencies among the five annotators. We estimate the dependencies $\Gamma_{rl} \in [0, 1]$ between the r-th and l-th annotator from (3.5) as:

$$\Gamma_{rl} = \langle \boldsymbol{Q}_r \bar{\boldsymbol{K}}_r \boldsymbol{Q}_r, \boldsymbol{Q}_l \bar{\boldsymbol{K}}_l \boldsymbol{Q}_l \rangle_{\mathrm{F}} \tag{3.14}$$

Similarly, we consider the absolute value of the Pearson correlation coefficient between the annotations $\boldsymbol{y}_r$ and $\boldsymbol{y}_l$ for $r, l \in \{1, \ldots, R\}$ as the reference value. Comparing the reference and the estimated dependencies among the annotators, we can see that, although the exact behavior is not recovered, LKAAR is able to identify the critical relationship among the labelers from the input features in $\boldsymbol{X}$. On the other, regarding semi-synthetic data results, Table 3.3(a) shows

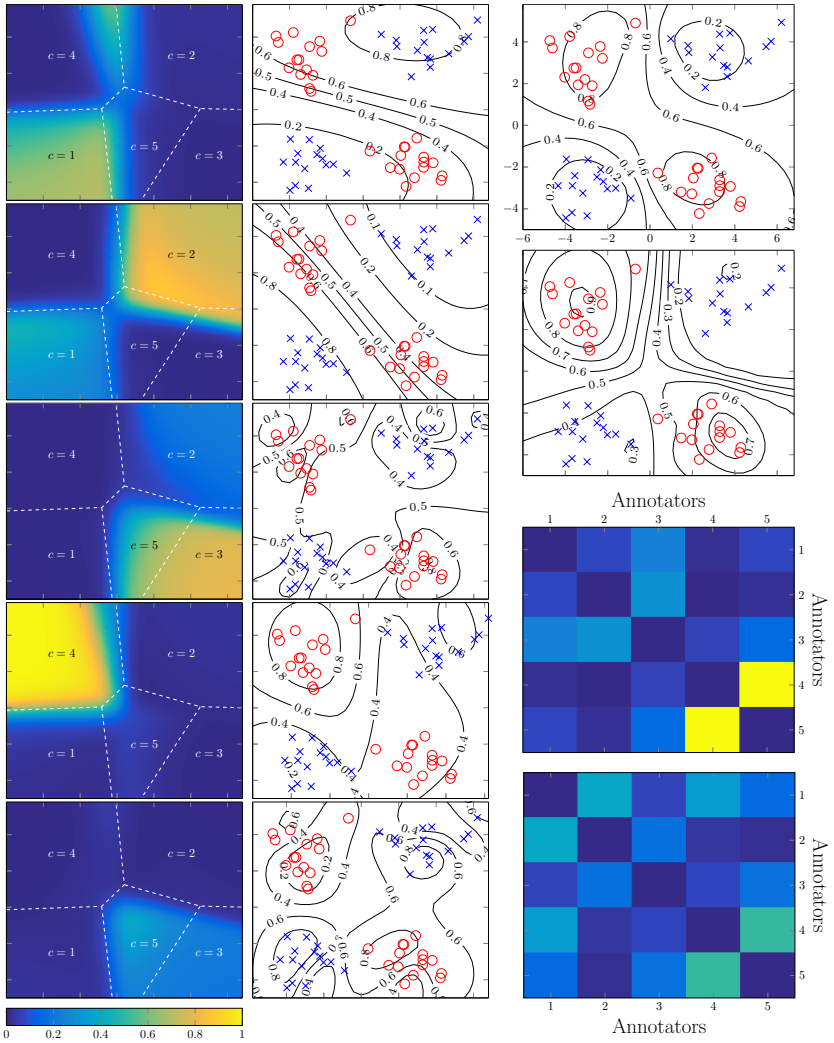

Figure 3.1 Fully synthetic dataset results. The first column (top to bottom): relevance values $q_r^2(\boldsymbol{x}_n)$. The second column (top to bottom): decision boundaries produced by a GPC trained over each annotator dataset. The third column (top to bottom): decision boundaries generated by the GPC-GOLD (gold standard) and the LKAAR. Also, it displays the dependencies among the annotators estimated by LKAAR (from the input samples) vs. the Pearson correlation coefficients (absolute value) from the labelers' annotations

the results regarding the "Biased coin" simulated labels, where the performance of the experts is constant across the input space. On average, KAAR exhibits the best generalization performance in AUC and Acc. However, we also note that our LKAAR-GPC gets the second-best performance in both metrics. The above is a satisfying result because both approaches are based on the combination of R classifiers, where combinations coefficients are proportional to the labeler's performance. Note that KAAR was designed assuming that the labelers' performance is constant for every region in the input space. Now, concerning other competitors based on non-linear classifiers, we note that the approach based on majority voting gets a low performance, which is not unexpected since this method is the most naive to deal with multi-labelers scenarios. Besides, we note that the behavior of MA-GPC is surprising since its performance is poor compared with other non-linear classifiers (KAAR, LKAAR-GPC, LKAAR-SVM). Linear clas-

Table 3.3 UCI repository classification results

(a) Biased coin labels

Method		breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	average
GPC-GOLD	AUC[%]	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	97.65 ± 2.71	99.22 ± 0.67	90.08 ± 1.94	92.12
	Acc[%]	96.44 ± 1.54	68.48 ± 4.43	91.08 ± 2.41	76.71 ± 1.96	99.16 ± 0.85	95.85 ± 3.29	96.92 ± 1.44	70.68 ± 6.81	86.91
GPC-MV	AUC[%]	89.57 ± 7.77	55.15 ± 9.03	67.71 ± 7.56	58.33 ± 5.13	61.94 ± 4.65	98.82 ± 1.15	97.01 ± 2.29	89.93 ± 2.27	77.31
	Acc[%]	84.31 ± 9.85	53.78 ± 7.34	63.78 ± 6.72	59.26 ± 4.43	59.72 ± 4.13	85.78 ± 5.59	86.29 ± 5.22	77.43 ± 5.99	71.29
MA-LFC	AUC[%]	98.83 ± 1.12	**69.92 ± 4.63**	82.91 ± 4.63	82.73 ± 2.40	59.67 ± 2.98	99.34 ± 0.77	97.63 ± 1.87	**99.50 ± 0.22**	86.32
	Acc[%]	95.80 ± 1.27	**65.96 ± 5.28**∗	83.43 ± 4.12	76.35 ± 2.10	66.82 ± 2.19	93.85 ± 4.66	90.13 ± 4.82	**94.89 ± 0.75**	83.40
MA-DGRL	AUC[%]	**99.23 ± 0.45**	64.03 ± 8.48	79.83 ± 5.36	82.89 ± 2.42	58.52 ± 3.04	98.88 ± 1.18	95.27 ± 2.81	98.42 ± 0.36	84.63
	Acc[%]	95.35 ± 1.45	60.80 ± 7.78	82.63 ± 3.72	77.32 ± 2.48	63.69 ± 2.54	92.74 ± 4.21	84.28 ± 6.20	89.47 ± 1.10	80.79
MA-MAE	AUC[%]	97.49 ± 3.57	65.89 ± 7.30	64.03 ± 9.02	**83.49 ± 2.25**	50.58 ± 6.34	89.12 ± 5.91	93.14 ± 5.78	95.69 ± 1.03	79.93
	Acc[%]	94.03 ± 3.07	64.01 ± 7.50	70.79 ± 9.96	**77.00 ± 2.67**∗	51.56 ± 8.00	78.74 ± 11.07	81.70 ± 8.82	90.99 ± 2.92	76.10
MA-GPC	AUC[%]	94.45 ± 2.51	43.19 ± 9.84	92.03 ± 6.05	68.96 ± 5.86	73.42 ± 4.15	99.09 ± 0.81	**99.73 ± 0.38**	99.33 ± 0.14	83.78
	Acc[%]	96.24 ± 1.02	54.23 ± 4.34	80.57 ± 6.34	73.33 ± 2.47	71.39 ± 3.32	93.85 ± 3.07	93.84 ± 3.17	93.95 ± 0.92	82.18
KAAR	AUC[%]	99.05 ± 0.60	67.14 ± 6.50	**93.73 ± 2.36**	81.34 ± 2.49	**91.32 ± 3.60**	**99.59 ± 0.64**	99.16 ± 0.60	92.10 ± 1.26	**90.43**
	Acc[%]	95.89 ± 1.01	63.91 ± 5.12	**89.46 ± 2.55**∗	76.00 ± 2.48	**84.73 ± 3.07**∗	**95.11 ± 3.16**∗	96.16 ± 2.60	82.25 ± 5.16	**85.44**∗
LKAAR-LR	AUC[%]	97.35 ± 2.28	60.71 ± 7.25	69.36 ± 8.28	76.18 ± 3.22	54.02 ± 3.53	95.53 ± 2.91	98.05 ± 1.30	98.23 ± 0.28	81.18
	Acc[%]	89.53 ± 5.34	57.21 ± 6.64	69.43 ± 7.78	69.99 ± 3.32	55.02 ± 3.54	85.85 ± 5.76	92.70 ± 3.96	90.93 ± 1.02	76.33
LKAAR-SVM	AUC[%]	98.90 ± 0.52	56.43 ± 3.84	92.79 ± 4.64	71.92 ± 3.69	76.98 ± 3.73	99.13 ± 1.16	99.50 ± 0.61	98.54 ± 0.26	86.77
	Acc[%]	**96.50 ± 1.10**∗	55.10 ± 2.97	88.63 ± 4.11	68.81 ± 3.19	71.71 ± 3.84	90.89 ± 7.62	95.91 ± 2.33	92.67 ± 1.12	82.53
LKAAR-GPC	AUC[%]	99.11 ± 0.69	59.54 ± 5.93	91.88 ± 3.08	79.57 ± 3.17	89.07 ± 2.93	99.57 ± 0.55	99.36 ± 0.57	92.76 ± 1.71	88.86
	Acc[%]	95.84 ± 1.46	56.60 ± 4.50	87.59 ± 2.80	75.06 ± 2.42	82.85 ± 3.13	95.26 ± 3.07	**96.48 ± 2.61**∗	82.58 ± 4.68	84.03

(b) Non-homogeneous labels

Method		breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	average
GPC-GOLD	AUC[%]	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	97.65 ± 2.71	99.22 ± 0.67	90.08 ± 1.94	92.12
	Acc[%]	96.44 ± 1.54	68.48 ± 4.43	91.08 ± 2.41	76.71 ± 1.96	99.16 ± 0.85	95.85 ± 3.29	96.92 ± 1.44	70.68 ± 6.81	86.91
GPC-MV	AUC[%]	99.11 ± 0.58	70.95 ± 3.90	93.14 ± 3.49	81.21 ± 2.57	87.83 ± 4.11	99.63 ± 0.39	98.41 ± 1.38	91.48 ± 1.48	90.22
	Acc[%]	96.29 ± 1.48	66.60 ± 4.31	87.90 ± 3.26	74.87 ± 2.32	81.96 ± 3.46	95.33 ± 3.05	93.96 ± 3.34	82.68 ± 5.30	84.95
MA-LFC	AUC[%]	98.72 ± 0.93	**71.53 ± 4.18**	82.08 ± 4.79	**82.29 ± 2.22**	61.13 ± 3.28	98.75 ± 1.44	96.83 ± 1.75	**99.58 ± 0.11**	86.36
	Acc[%]	95.63 ± 1.79	**69.68 ± 4.20**∗	81.43 ± 4.44	**76.52 ± 1.91**∗	64.88 ± 2.86	94.44 ± 4.62	87.74 ± 4.67	**95.40 ± 0.71**∗	83.21
MA-DGRL	AUC[%]	99.30 ± 0.39	68.00 ± 4.09	77.60 ± 7.50	81.72 ± 2.57	61.83 ± 2.80	98.78 ± 1.34	95.33 ± 3.35	98.31 ± 0.32	85.11
	Acc[%]	94.63 ± 1.77	65.77 ± 3.47	81.94 ± 3.42	76.45 ± 2.81	66.45 ± 2.24	94.59 ± 2.96	84.91 ± 6.50	89.58 ± 0.99	81.79
MA-MAE	AUC[%]	99.28 ± 0.60	70.82 ± 3.90	78.91 ± 6.01	81.80 ± 2.57	60.35 ± 3.28	85.97 ± 2.39	98.20 ± 1.33	97.27 ± 0.28	85.32
	Acc[%]	96.31 ± 1.38	66.92 ± 3.35	82.25 ± 3.99	76.12 ± 2.77	65.64 ± 2.37	94.81 ± 4.14	89.31 ± 5.79	92.94 ± 0.76	83.04
MA-GPC	AUC[%]	95.81 ± 2.94	49.81 ± 11.72	94.46 ± 3.09	67.83 ± 4.24	81.44 ± 3.81	99.15 ± 1.03	**99.85 ± 0.24**	99.42 ± 0.14	85.97
	Acc[%]	**96.70 ± 1.37**∗	59.52 ± 4.71	82.13 ± 3.32	72.77 ± 2.71	76.39 ± 2.85	94.30 ± 2.90	94.34 ± 2.80	94.74 ± 0.68	83.86
KAAR	AUC[%]	98.81 ± 0.66	70.20 ± 5.70	93.88 ± 3.53	81.18 ± 2.93	89.55 ± 2.84	99.56 ± 0.52	99.53 ± 0.36	92.34 ± 1.38	90.63
	Acc[%]	96.02 ± 1.14	65.99 ± 5.44	87.52 ± 4.24	75.10 ± 2.98	81.68 ± 2.41	**95.56 ± 2.92**∗	**96.54 ± 2.11**∗	81.11 ± 4.15	84.94
LKAAR-LR	AUC[%]	**99.34 ± 0.44**	68.86 ± 5.16	87.14 ± 3.38	82.04 ± 2.44	65.40 ± 3.13	96.00 ± 2.50	99.21 ± 0.82	97.97 ± 0.27	87.00
	Acc[%]	96.00 ± 1.46	64.17 ± 4.22	84.10 ± 3.20	75.67 ± 2.15	66.96 ± 2.74	82.59 ± 6.07	94.28 ± 3.19	90.02 ± 0.93	81.72
LKAAR-SVM	AUC[%]	98.29 ± 0.80	64.37 ± 3.36	**96.98 ± 2.01**	77.80 ± 2.28	89.82 ± 2.14	98.05 ± 1.90	99.53 ± 0.47	97.89 ± 0.32	90.34
	Acc[%]	96.36 ± 1.02	63.14 ± 3.67	**92.19 ± 2.43**∗	72.52 ± 2.22	80.99 ± 2.81	84.44 ± 6.76	96.48 ± 2.26	91.28 ± 0.93	84.68
LKAAR-GPC	AUC[%]	99.00 ± 0.75	71.07 ± 5.05	93.37 ± 2.91	81.23 ± 2.21	**91.97 ± 2.01**	99.57 ± 0.61	99.64 ± 0.34	92.61 ± 1.73	**91.06**
	Acc[%]	96.03 ± 1.32	66.92 ± 4.79	87.75 ± 3.90	75.10 ± 2.65	**84.09 ± 2.43**∗	95.26 ± 3.29	**96.54 ± 2.16**∗	80.98 ± 3.91	**85.33**∗

(c) Biased coin (Non-homogeneous) labels

Method		breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	average
GPC-GOLD	AUC[%]	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	97.65 ± 2.71	99.22 ± 0.67	90.08 ± 1.94	92.12
	Acc[%]	96.44 ± 1.54	68.48 ± 4.43	91.08 ± 2.41	76.71 ± 1.96	99.16 ± 0.85	95.85 ± 3.29	96.92 ± 1.44	70.68 ± 6.81	86.91
GPC-MV	AUC[%]	90.78 ± 4.28	50.47 ± 6.19	82.91 ± 6.03	70.18 ± 6.29	65.91 ± 6.72	98.55 ± 1.38	97.75 ± 2.04	90.18 ± 1.71	80.84
	Acc[%]	86.63 ± 2.06	48.27 ± 4.84	75.65 ± 6.45	66.52 ± 5.16	64.66 ± 3.64	88.81 ± 5.00	86.92 ± 5.76	79.24 ± 4.99	74.59
MA-LFC	AUC[%]	97.99 ± 0.99	59.64 ± 8.08	72.66 ± 9.98	72.73 ± 3.43	52.88 ± 3.13	96.72 ± 8.98	96.47 ± 2.13	**99.50 ± 0.15**	81.07
	Acc[%]	**96.00 ± 1.70**∗	56.41 ± 8.12	69.17 ± 12.53	58.10 ± 4.53	46.27 ± 3.03	92.30 ± 5.18	87.55 ± 4.97	**95.06 ± 0.80**∗	75.11
MA-DGRL	AUC[%]	99.31 ± 0.42	61.77 ± 6.17	77.83 ± 7.02	**81.66 ± 2.65**	55.70 ± 3.95	98.76 ± 1.33	95.26 ± 3.30	98.32 ± 0.34	83.58
	Acc[%]	78.08 ± 2.22	55.64 ± 4.52	71.43 ± 5.15	**76.90 ± 1.99**∗	60.64 ± 2.33	94.37 ± 2.66	84.84 ± 6.32	89.63 ± 0.89	76.44
MA-MAE	AUC[%]	95.22 ± 1.70	64.63 ± 9.77	64.18 ± 9.17	79.94 ± 2.64	52.36 ± 4.78	93.16 ± 5.08	96.25 ± 2.40	94.40 ± 1.26	80.02
	Acc[%]	87.15 ± 1.85	**62.34 ± 8.46**∗	67.94 ± 7.19	75.94 ± 2.69	53.33 ± 6.42	81.70 ± 11.68	86.67 ± 5.15	88.38 ± 2.00	75.43
MA-GPC	AUC[%]	85.37 ± 5.90	40.79 ± 12.30	74.52 ± 4.57	73.17 ± 3.34	61.82 ± 4.51	98.71 ± 1.14	**99.60 ± 0.41**	99.35 ± 0.14	79.17
	Acc[%]	92.55 ± 2.17	52.82 ± 6.38	69.87 ± 4.41	62.42 ± 3.00	62.33 ± 2.98	93.85 ± 3.49	**95.09 ± 2.65**∗	93.46 ± 0.83	77.80
KAAR	AUC[%]	97.81 ± 0.99	56.52 ± 9.13	82.20 ± 4.93	67.90 ± 3.16	75.34 ± 4.70	98.75 ± 1.10	97.91 ± 1.36	91.75 ± 1.41	83.52
	Acc[%]	77.19 ± 3.14	52.44 ± 7.79	72.60 ± 4.80	61.20 ± 2.95	70.69 ± 3.63	90.44 ± 5.48	91.45 ± 4.28	76.38 ± 5.05	74.05
LKAAR-LR	AUC[%]	**99.52 ± 0.30**	**66.07 ± 6.14**	82.99 ± 5.01	80.57 ± 3.31	52.32 ± 3.38	96.83 ± 2.14	99.27 ± 0.68	97.87 ± 0.30	84.43
	Acc[%]	92.47 ± 2.24	60.22 ± 5.67	78.92 ± 4.32	75.07 ± 2.65	55.64 ± 2.77	83.41 ± 6.92	94.59 ± 3.12	89.77 ± 0.99	78.76
LKAAR-SVM	AUC[%]	98.37 ± 1.00	52.35 ± 6.40	**88.28 ± 5.13**	66.84 ± 3.66	73.85 ± 3.43	96.22 ± 2.50	98.88 ± 0.80	97.59 ± 0.34	84.05
	Acc[%]	87.72 ± 5.17	50.96 ± 6.81	**84.73 ± 4.66**∗	64.81 ± 3.11	70.02 ± 2.74	74.15 ± 7.90	91.82 ± 4.33	90.37 ± 1.24	76.82
LKAAR-GPC	AUC[%]	98.14 ± 1.04	58.36 ± 7.24	86.23 ± 4.47	73.80 ± 2.83	**80.02 ± 4.15**	**99.61 ± 0.61**	98.74 ± 0.93	92.24 ± 1.80	**85.89**
	Acc[%]	86.76 ± 4.33	54.52 ± 5.27	78.25 ± 5.51	69.64 ± 3.01	**74.90 ± 2.99**∗	**95.93 ± 3.15**∗	93.84 ± 3.57	78.71 ± 4.18	**79.07**∗

Bold: the highest AUC excluding the upper bound (target) classifier GPC-GOLD. The highest accuracy (Acc) is marked with ∗, except the upper bound.

sifiers such as MA-LFC and MA-DGRL obtain a better performance. This result indicates that MA-GPC is more prone to overfitting, which authors in [19] empirically analyzed. Next, we an-

alyze the results concerning linear classifiers. Notably, simple classifiers such as MA-LFC and MA-DGRL obtain competitive results compared with their non-linear competitors. The above suggests that some of the considered datasets may have a linear structure. To confirm this supposition, we perform an additional experiment by training an LR-based classifier with the actual labels over all the datasets (we follow the same scheme as for GPC-GOLD). We obtain an AUC equal to 87.10 (on average), close to the results of MA-LFC and MA-DGRL. We can elucidate that there exists a linear structure in some of the datasets, and under such settings, MA-LFC, and MA-DGRL sets an attractive option. Nonetheless, MA-MAE and LKAAR-LR obtain the worst generalization performance (MA-MAE performs worse than GPC-MV), which is a clear consequence of their feature-dependent model for the annotators. We recall that the process of generating the synthetic annotations assumes that the annotators' performance does not depend on the input features; hence, this scenario is most convenient for MA-LFC and MA-DGRL.

On the other hand, Table 3.3(b) shows the results regarding the "Non-homogeneous labels", where the performance of the experts depends on the input features. First, we observe that all classification models exposed better generalization performance in AUC and Acc compared with the previous simulation method (Table 3.3(a)). The above suggests an increase in the quality of the labels; in fact, we can confirm this suggestion by analyzing the behavior of GPC-MV, which gets quite competitive results. Besides, we highlight that our LKAAR-GPC exhibits, on average, the best performance among the nine multi-annotators classifiers (in terms of AUC and Acc). The behavior of the non-linear based approaches is, in principle, a bit unexpected in that the most straightforward method (GPC-MV) reaches a comparable performance with more sophisticated strategies such as KAAR and LKAAR-SVM. However, as we already pointed out, this behavior is caused because, in this experimental set-up, the simulation method ("Non-homogeneous labels") generates suitable quality labels, which favors the estimation of the unknown ground truth via majority voting. Besides, MA-GPC again gets a considerably low performance compared with its other non-linear based competitors; still, this can result from a lack of generalization (overfitting). Now, let us analyze the results for the linear models. Remarkably, we notice that LKAAR-LR and MA-MAE get more competitive results when compared with their competitors MA-LFC and MA-DGRL. Since the labelers' expertise is non-stationary across the input space, both LKAAR-LR and MA-MAE are suitable. Furthermore, we highlight that our LKAAR-LR outperforms all linear competitors, and its generalization performance is quite better than its natural competitor MA-MAE.

Finally, Table 3.3(c) shows the results as regards the experiment where the labels were simulated by using the method "Biased coin (Non-homogeneous)". At first glance, this experiment is more challenging since there exists more difference between the performance of GPC-GOLD (the upper bound) and the classification scheme with the best performance (our LKAAR-GPC). Analyzing the non-linear classifiers, we note that again, MA-GPC obtains an insufficient performance, which is even worse than the lower bound (GPC-MV). Besides, we remark the

Table 3.4 Fully real datasets results

AUC([%])						
Method	**Voice Dataset**			**Polarity Dataset**	**Music**	**Average**
	G	**R**	**B**			
GPC-GOLD	93.66	93.66	93.66	80.26	92.84	90.81
GPC-MV	90.17	84.73	84.04	71.14	89.20	83.97
MA-LFC	89.99	90.59	87.27	72.06	89.63	85.90
MA-DGRL	85.45	90.14	79.33	74.35	89.20	83.69
MA-MAE	91.08	89.12	80.74	50.00	84.16	79.02
MA-GPC	91.50	91.16	80.81	**77.18**	79.18	83.96
KAAR	89.85	93.50	89.20	64.68	86.74	84.79
LKAAR-LR	90.39	92.92	88.94	76.99	89.36	87.72
LKAAR-SVM	**92.06**	93.02	86, 98	75.68	**90.79**	87.70
LKAAR-GPC	90.78	**93.60**	**89.79**	77.16	90.69	**88.40**

Bold: the method with the highest performance excluding the upper bound (target) classifier GPC-GOLD.

performance of our LKAAR-SVM and LKAAR-GPC, which outperform all of its competitors (GPC-MV, KAAR, and MA-GPC). The latter is an exciting result, especially when we compare the KAAR and LKAAR-GPC performances (both approaches are based on the combination of GPCs). Concerning the linear classifiers, we remark that our LKAAR-LR surpasses all its competitors (MA-LFC, MA-DGRL, MA-MAE). Indeed, our approach obtained the second-best results, also defeating non-linear classifiers such as GPC-MV, KAAR, and MA-GPC. The latter is evidence that our approach better represents the annotators' behavior. Besides, we notice that the performance of MA-MAE is quite low compared with our LKAAR-LR, which is, in principle, unexpected since both approaches compute the annotators' performance as a function of the input space. This result can be explained in two regards. First, unlike our LKAAR-LR, MA-MAE uses a logistic regression-based model to code the non-stationaries in the labelers' performance, which does not fit the labels generated in this experiment. Moreover, MA-MAE assumes independence between the annotators; the labelers make their decision independently, which decreases the modeling of the labeler's behavior.

Summarizing, we tested our approach in controlled scenarios by using three different strategies. First, we simulate annotators with homogeneity in their performance named *Biased coin labels*. The remaining two strategies named *Non-homogeneous labels* and *Biased coin (Non-homogeneous) labels* simulates inconsistent annotators, *i.e.*, labelers, whose performance varies depending on the input features. Attained to the results (Table 3.3(a), Table 3.3(b), and Table 3.3(c)), we note that for consistent annotators KAAR offers the best performance. On the other hand, for inconsistent labelers, our LKAAR is the best option. However, we highlight that by considering inconsistent annotators, a more realistic scenario is configured [1, 50, 52]; hence, our approach is presented as the most suitable option among the state-of-the-art models considered.

Until now, we have empirically demonstrated that our approach offers a better representation of the labelers' behavior since we compute the annotators' performance taking into account

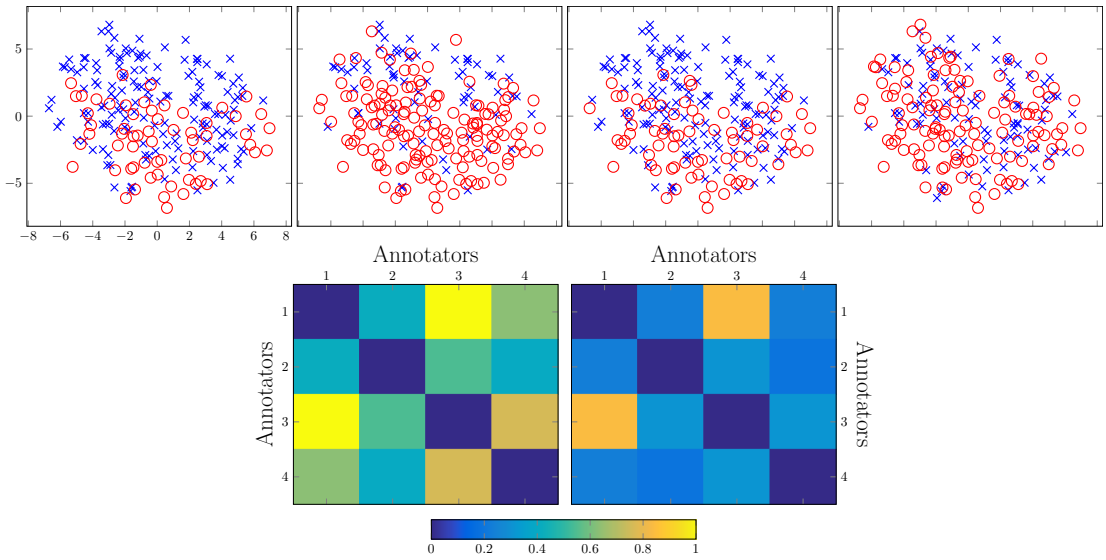

Figure 3.2 Annotator dependencies analysis for the Voice dataset (Scale B). First row (left to right): t-sne-based 2D projections holding the annotators' labels. Red circles: positive class, blue crosses: negative class. Second row (left to right): dependencies among annotators estimated by LKAAR (from the input samples) vs. the Pearson correlation coefficients (absolute value) from the experts' labels

dependencies among their decisions and considering that such a performance as a function of the input features. Nevertheless, the previous experiments configure a very controlled scenario due to the labels are simulated. Accordingly, these results could be biased by the simulation method. In this sense, the *fully real datasets* present the the most challenging scenario, where both the input samples and the labels come from real-world applications. Table 3.4 outlines the achieved performances. First, we observe that for the voice data, the scales **G** and **R** exhibit a similar performance for all studied approaches. The latter can be explained in that for these scales, the annotators exhibit a proper performance. On the other hand, we notice that for the scale **B**, there exits a considerable performance reduction, which indicates that the annotators' performance is lower when compared with scales **G** and **R**, as was empirically demonstrated in [21].

However, we highlight that our approach (LKAAR-LR, LKAAR-SVM, LKAAR-GPC) and KAAR exhibit the best generalization performances. Remarkably, LKAAR-LR outperforms all of its linear competitors. This is an outstanding outcome because it reflects that our approach better represents the labelers' behavior. We perform an additional experiment over the voice dataset for the scale B. For visualizations purposes; we reduce the dimensions to 2D by using the well-known t-student stochastic neighbor embedding algorithm (t-sne). From left to right, Figure 3.2 (row 1) shows each expert's labels on the 2D t-sne space. Likewise, in the second row, from left to right, we show the LKAAR estimation of the dependencies among the annotators vs. the absolute value of the Pearson correlation coefficient between $\boldsymbol{y}_r$ and $\boldsymbol{y}_l$, for $r, l \in \{1, \ldots, R\}$. As seen, the labels provided by the first and third annotator are quite similar, indicating a strong dependence between them. Remarkably, this structure in the annotation process is captured by LKAAR.

Next, regarding the Polarity dataset, we note that MA-GPC exhibits the best generalization performance followed by our LKAAR-GPC; the difference between these two methods is not significant. Besides, the performance of MA-MAE and KAAR is quite low, in principle, unexpected; however, we give some reasons to explain this anomaly. First, MA-MAE uses P parameters for the classification model and $P \times R$ parameters to model the labelers' performance. Moreover, from Table 2.1, we can note that for this dataset, $P = 1200$ and the number of training samples is 5000. The above is a problem because, considering that we have labels from 203 annotators, we need to estimate over 240000 parameters, leading MA-MAE to overfit. Now, the behavior of KAAR is more unexpected than MA-MAE since it is based on a non-linear classifier and exhibited a competitive performance in previous experiments. We argue that the missing labels cause this unusual conduct because more than the 50% of the annotators labeled less than 30 instances. The above can represent a problem for KAAR because it estimates the labelers' performance as an average matching between the input features and the labels each annotator gives. Accordingly, the labelers' performance estimation suffers a negative impact in scenarios with very few labels per annotator. We will expand this analysis in the final experiment.

Lastly, we analyze the results concerning Music dataset. We remark that our LKAAR-SVM and LKAAR-GPC obtain the best generalization performance in AUC. Moreover, we highlight the achievement of our LKAAR-LR, given that it gets a competitive result compared with the other linear classifiers, even defeating some non-linear classifiers (KAAR, MA-GPC, and GPC-MV). On the other hand, we note that MA-MAE and MA-GPC exhibit a significantly low performance, even lower than their intuitive lower bound (GPC-MV). This behavior is not uncommon, given that it has been repeated in the previous experiment; we argue that this outcome results from overfitting. However, this dataset configures a multi-class classification problem; accordingly, we use a *one-vs-all* scheme for all binary classification (including MA-MAE and MA-GP) to deal with this. Nevertheless, such a scheme to deal with multi-class classification can lead to regions on the input space that are ambiguously classified [9].

As a final experiment, we wish to evaluate the impact of the number of annotators on the performance of the multi-labeler classifiers. We use the Music dataset for concrete testing, which holds 2946 labeled instances from 44 real annotators. We sort the experts in a descending way regarding the number of cases marked. Figure 3.3 shows the number of samples labeled by each annotator. We can note that only a few annotators labeled more than 300 samples; more than 50% of the labelers annotated less than 30 examples.

Figure 3.4 shows the classifiers' performance in AUC as a function of the number of annotators. First, we notice that our LKAAR-SVM and LKAAR-GPC exhibit, on average, the best performances in terms of AUC (89.53, 87.60, respectively). Both approaches have no significant effects due to the number of annotators. In turn, we note an unusual behavior in MA-GPC and MA-MAE. Such action is caused by overfitting. On the other hand, we analyze that the linear models MA-LFC, LKAAR-LR, MA-GRL, and GPC-MV expose quite similar behavior, where

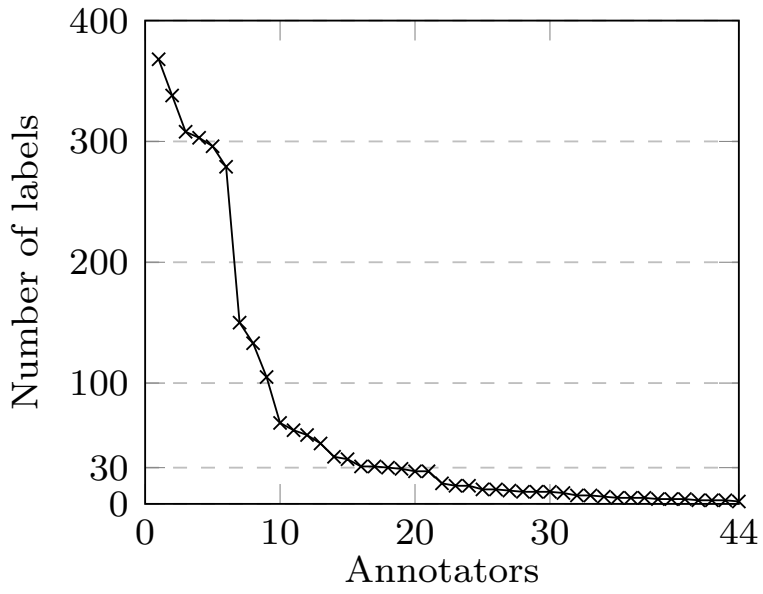

Figure 3.3 Number of labeled instances per annotator

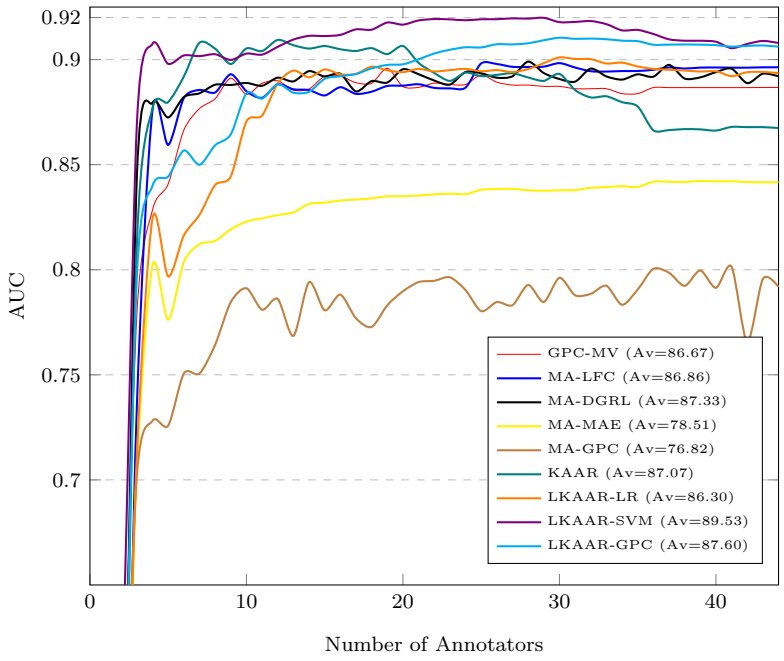

Figure 3.4 Generalization performance in terms of AUC as a function of the number of annotators

their performance is not affected by the number of annotators. KAAR exhibits a suitable performance; however, after 20 annotators, we evidence a significant decrease in its execution; indeed, for 44 labelers, KAAR performs worse than linear classifiers. We can explain this result as follows: The performance of KAAR starts to decrease coincides with the point where the number of labeled instances per annotator decreases. Accordingly, from the annotator 20, KAAR has to estimate the experts' performance with very few labels, and this could lead to miss-estimation, which harms the predictions.

3.4.2 Regression

Following a similar structure for the classification results, we first conduct a controlled experiment to verify the LKAAR capabilities in the context of real-valued labels. For this first experiment, we use the *fully synthetic* dataset described in Section 3.3.1. We simulate five labelers ($R = 5$) with different levels of expertise and using the following variance matrix $\boldsymbol{V} \in \mathbb{R}^{R\times R}$:

$$\boldsymbol{V} = \begin{bmatrix} 0.00 & 3.00 & 0.02 & 2.00 & 1.50 \\ 1.00 & 0.00 & 2.00 & 1.50 & 0.02 \\ 0.75 & 1.50 & 0.00 & 0.02 & 1.00 \\ 0.50 & 0.02 & 1.00 & 0.00 & 2.00 \\ 0.02 & 2.00 & 3.00 & 1.00 & 0.00 \end{bmatrix}, \tag{3.15}$$

holding elements $\sigma_{c,r}$, which are related to the expertise of the r-th annotator in the region c. We notice that the r-th annotator is an expert (its labels correspond to the ground truth) in the region $k = r$. Figure 3.5 column 1 and 2 present respectively the regression functions for each annotator g_r and the LKAAR-based weights $q_r^2(\boldsymbol{x}_n)$, related to the labelers' performance. Contrasting the parameters for the simulation process (equation (3.15)) and the LKAAR-based results in the second column, we can note that our approach can detect the zones where the parameter $\sigma_{c,r}$ presents the lowest value and remarkably such zones match with regions where the regression results are closer to the gold standard (the above from a visual inspection). Further, in the first two images (from top to bottom) of the third column of Figure 3.1, we show the result of GPC-GOLD (a Gaussian processes regressor trained with the gold standard) and the regression results generated by our LKAAR-based approach as in equation (3.10). We notice that compared with the ground truth, our approach exhibit a proper performance; in fact, the result for our LKAAR is close to its theoretical upper bound GPC-GOLD. By estimating the $q_r(\boldsymbol{x}_n)$ weights as a function of the input space, we can enhance the zones where the regression function for r-th labeler, g_r, exhibits high performances. Besides, one of the main aims of this proposal is to estimate the annotators' performance by considering inter-dependencies. Therefore, in the last two images of the third column in Figure 3.5 we expose a visual comparison between the estimated dependencies among the five annotators.

We estimate the dependencies $\Gamma_{rl} \in [0, 1]$ between the r-th and l-th annotator from (3.5). Likewise, we consider the absolute value of the Pearson correlation coefficient between the annotations $\boldsymbol{y}_r$ and $\boldsymbol{y}_l$ for $r, l \in \{1, \ldots, R\}$ as the reference value. Comparing the reference and the estimated dependencies with LKAAR, we remark that our approach identifies critical relationships among thelabelers.

On the other hand, Table 3.5 presents the results concerning semi-synthetic datasets. We highlight that our LKAAR exhibits the best regression performance in terms of the R^2 score. Now, analyzing the behavior of GPs-based competitors, we notice that GPR-Av exhibits a lower performance when compared with our KAAR, which is not unexpected since GPR-Av corre-

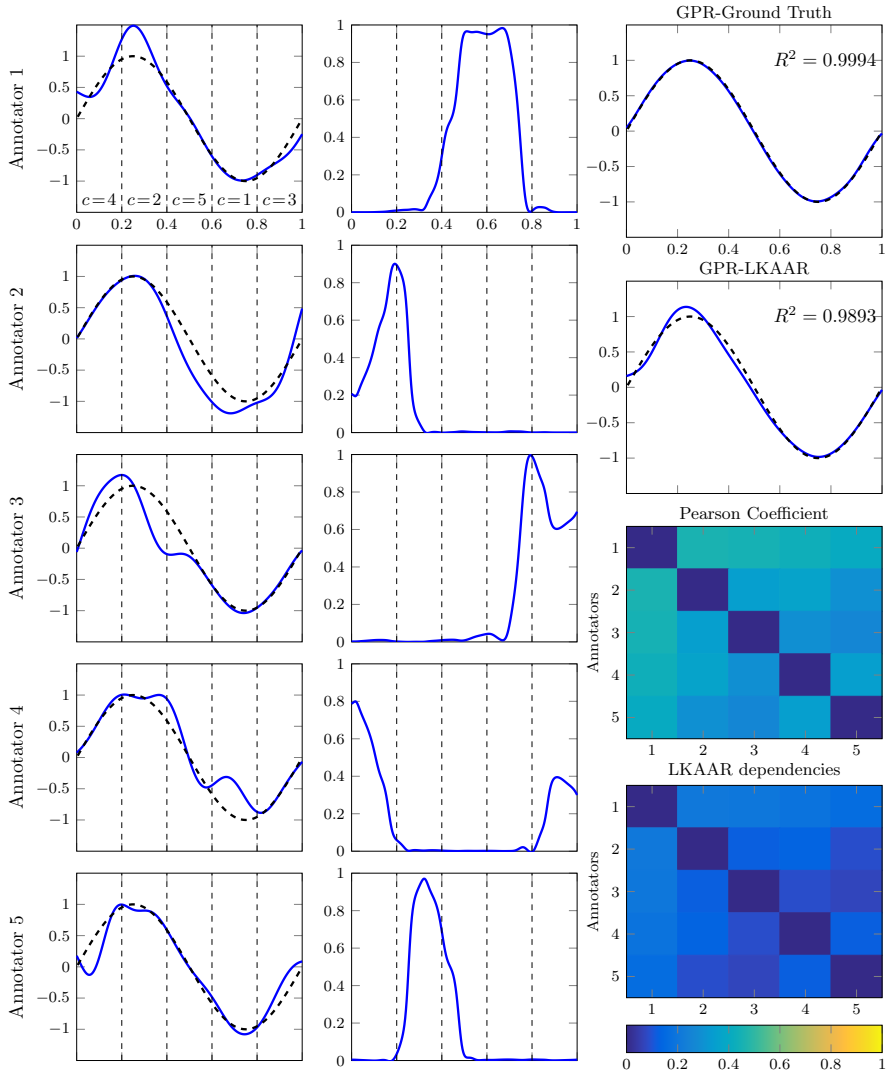

Figure 3.5 Regression illustrative results. In the first column, from top to bottom, we show the regression results g_r for each annotator. On the second column, from top to bottom, we present the LKAAR-based relevance q_r. In the third column, we exhibit the result for GPR-GOLD and the result for our GPR-LKAAR; finally, in the last row, we show the Pearson coefficient computed over the labels from multiple annotators and the dependencies estimated with our LKAAR

sponds to the most naive approach to deal with multi-labeler data. Besides, we remark a considerably low R^2 score (on average) for MA-GPR, even lower than the supposed lower bound

Table 3.5 Regression results in terms of R^2 score over *semi-synthetic datasets*

Method	auto	Bike	concrete	housing	yacht	ct	Average
GPR-GOLD	0.8698 ± 0.0268	0.5612 ± 0.0059	0.8101 ± 0.0251	0.8288 ± 0.0389	0.8043 ± 0.0741	0.8652 ± 0.0054	0.7899
GPR-Av	0.8480 ± 0.0348	0.5479 ± 0.0069	0.7471 ± 0.1452	0.4972 ± 0.4034	0.7368 ± 0.1665	0.8261 ± 0.0079	0.7005
MA-LFCR	0.7975 ± 0.0230	0.3880 ± 0.0072	0.5965 ± 0.0369	0.7033 ± 0.0570	0.6065 ± 0.0834	0.6605 ± 0.2503	0.6254
MA-GPR	$\mathbf{0.8558 \pm 0.0269}$	0.4242 ± 0.0175	0.7807 ± 0.0312	0.7247 ± 0.0628	0.7319 ± 0.0965	0.0111 ± 0.0038	0.5881
MA-DL-B	0.7814 ± 0.0326	$\mathbf{0.5914 \pm 0.0131}$	0.2228 ± 0.0361	0.5351 ± 0.0781	0.1924 ± 0.0767	0.5477 ± 0.2991	0.4785
MA-DL-S	0.7835 ± 0.0302	0.5871 ± 0.0127	0.2186 ± 0.0328	0.5284 ± 0.0867	0.1809 ± 0.0976	$\mathbf{0.9402 \pm 0.0284}$	0.5398
MA-DL-B+S	0.7876 ± 0.0285	0.5877 ± 0.0116	0.2270 ± 0.0344	0.5339 ± 0.0881	0.1921 ± 0.0830	0.6431 ± 0.2999	0.4952
KAAR	0.8499 ± 0.0365	0.5672 ± 0.0056	0.7666 ± 0.0290	0.8037 ± 0.0448	0.5875 ± 0.0999	0.9051 ± 0.0023	0.7467
LKAAR	0.8484 ± 0.0377	0.5688 ± 0.0062	$\mathbf{0.7986 \pm 0.0246}$	$\mathbf{0.8047 \pm 0.0443}$	$\mathbf{0.7511 \pm 0.0875}$	0.8944 ± 0.0025	**0.7777**

Bold: the highest R^2 excluding the upper bound GPR-GOLD.

GPR-Av. Our explanation for such an outcome lies in the MA-GPR's formulation; MA-GPR models the annotators' behavior without considering relationships between the input space and the annotators' performance or dependencies.

Regarding the linear model, MA-LFR, we note that its prediction performance is considerably lower than our method. The above indicates the presence of a non-linear structure in most considered datasets. Finally, analyzing the DL-based methods, we remark that their performance is the lowest concerning the average R^2 score. Our explanation for this result is that the Crowd-layer used for the DL models to manage multi-labeler data does not offer a suitable representation of the annotators' behavior.

As a final experiment, we use the Music dataset corresponding to *fully real datasets*, configuring the most challenging scenario; here, both the input samples and the labels from multiple annotators come from real-world applications. Table 3.6 outlines the obtained performances in

Table 3.6 Regression results in terms of R^2 score over the *fully real dataset*

Method	Music
GPR-GOLD	0.4889
GPR-Av	0.2744
MA-LFCR	0.1404
MA-GPR	0.0090
MA-DL-B	0.2339
MA-DL-S	0.2934
MA-DL-B+S	**0.3519**
KAAR	0.2816
LKAAR	0.2865

Bold: the highest R^2 excluding the upper bound GPR-GOLD.

terms of the R^2 score. We remark that the DL-based approaches, MA-DL-B+S and MA-DL-S obtain the best generalization performance in terms of the R^2 score, followed by our LKAAR. Further, as theoretically expected, such performance lies between that of GPR-GOLD and GP-Av. Moreover, regarding the GPs-based competitor MA-GPR, we note that it exhibits the worst prediction capability with a R^2 close to zero. We argue that the above is a symptom of overfitting, as confirmed in Chapter 2. Besides, MA-LFCR presents the second-lowest performance, even worsen than the theoretical lower bound GPR-Av that suggests a non-linear structure in the music dataset. Finally, we notice a considerable reduction in the R^2 score compared with previous results (Table 3.5), because we are using regression schemes to solve a multi-class problem.

3.5 Summary

In this second proposal, we expose a localized kernel-alignment-based annotator relevance approach, named LKAAR, to support binary and multi-class classification problems in the pres-

ence of multiple annotators. Our LKAAR computes the relevance of each provided expert through a centered kernel alignment-based matching between the annotator labels and the input features, taking into account dependencies among the annotators and considering the labelers' performance as a function of the input data. Then, a combination of classifiers/regressors is carried out by adopting the multiple annotator performances coded in the LKAAR-based relevance analysis. We tested our approach in synthetic, semi-synthetic, and real-world datasets. For the synthetic experiments, we generate both the input data and the labels from multiple annotators. Similarly, we use some databases from the UCI repository for the semi-synthetic scenario. Regarding the real-world datasets, we include three datasets. The annotations are obtained from the opinion of multiple experts (voice quality problem) and using the crowd-sourcing platform AMT (Polarity data and Music). The results show that the proposed method deals with binary, multi-class classification, and regression problems, where the ground truth is not available. In most cases, our LKAAR achieves competitive or even better results when compared to different state-of-the-art classification and regression models [1, 20, 22, 27, 28, 54]. Besides, we empirically demonstrated that LKAAR better codes each annotator's behavior. The latter is also preserved for non-stationary labels across the input space (Figure 3.1). Moreover, Table 3.3 and Table 3.4, show that our approach is not severely affected by low labelers' performances. Besides, from Figure 3.3, we found that LKAAR exhibits a suitable generalization performance even after varying the number of annotators and the labeled samples, which is a remarkable output since it indicates that our proposal extracts relevant information from the labels, even if they are scarce. Even with KAAR, our LKAAR solves the supervised learning problem as a convex combination of classifiers/regressors (one per labeler), which can be problematic if the number of labelers grows. Future work must be oriented to develop an approach that jointly estimates the annotators' performance while training the supervised learning algorithm.

Chapter 4. Regularized Chained Deep Neural Network for Multiple Annotators

This chapter introduces a Regularized Chained Deep Neural Network for Multiple Annotators, RCDNN to jointly estimate the ground truth label and the annotators' performance. RCDNN is inspired in the Chained Gaussian Processes model–(CGP) [59], where each parameter in a given likelihood is coded with multiple independent Gaussian Processes–(GPs) priors (one GP prior per parameter). Unlike CGP, our method considers that the last layer models the parameters of an arbitrary likelihood. Thus, in a multi-labeler scenario, the annotators' parameters are coded as a function of the input space. Moreover, since each output in a deep model is computed as a linear combination of previous layers' outputs, our RCDNN can code interdependencies among the annotators. Besides, l1, l2, and Monte-Carlo Dropout-based regularizers are coupled within our method to deal with the overfitting issue in deep learning models. Our proposal follows the line of the works in [3, 25] in that RCDNN uses a deep-based approach to build a supervised learning model in the context of multiple annotators. However, while such approaches code the annotators' parameters as fixed points, we model them as functions to consider dependencies between the input features and the labelers' behavior. RCDNN is also similar to the LKAAR model introduced in Chapter 3. Both approaches model the annotators' performance as a function of the input instances and consider interdependencies among the labelers. Nonetheless, unlike LKAAR, where it is necessary to use as many classifiers as annotators, our approach only needs to train a single classifier from a deep learning representation, which is advantageous for many labelers.

4.1 Chained Deep Neural Network

Let us consider an input-output dataset $\mathcal{D} = \{\boldsymbol{X} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}\}$, where $\boldsymbol{X} = \left\{\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^P\right\}_{n=1}^N$ and $\boldsymbol{y} = \{y_n \in \mathcal{Y}\}_{n=1}^N$ hold the input and output spaces, respectively (with N instances and P features). Inspired by the Chained Gaussian Processes model–(CGP) [59], a likelihood function with J parameters is written as:

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} p(y_n|\theta_1(\boldsymbol{x}_n), \ldots, \theta_J(\boldsymbol{x}_n)), \tag{4.1}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \ldots, \boldsymbol{\theta}_J]^\top \in \mathbb{R}^{NJ}$ is a parameter vector, and $\boldsymbol{\theta}_j = [\theta_j(\boldsymbol{x}_1) \ldots \theta_j(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$. Here, each $\theta_j(\boldsymbol{x}) \in \mathcal{M}_j$ maps an input sample to the parameter space, being $\mathcal{M}_j$ the domain for the j-th parameter ($j \in \{1, 2, \ldots, J\}$). A Chained Deep Neural Network–(CDNN) can be introduced by linking each likelihood parameter $\theta_j(\boldsymbol{x})$ to one of the J outputs of a deep neural network comprising S hidden layers. Accordingly, let $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), \ldots, f_J(\boldsymbol{x})]^\top \in \mathbb{R}^J$ be a vector

containing the J outputs of a deep network:

$$\boldsymbol{f}(\boldsymbol{x}) = (\varrho_S \circ \varrho_{S-1} \circ \ldots \circ \varrho_1)(\boldsymbol{x}), \tag{4.2}$$

where $\circ$ stands for function composition. Then, each parameter is computed as: $\theta_j(\boldsymbol{x}) = h_j(f_j(\boldsymbol{x}))$, where $h_j : \mathbb{R} \to \mathcal{M}_j$ is a deterministic function that maps each output $f_j(\boldsymbol{x})$ to the appropriate domain $\mathcal{M}_j$. Besides, each layer ϱ_s, with $s \in \{1, 2, \ldots, S\}$, depends on a set of variables (neural network weights and bias) $\boldsymbol{\phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_S]^\top$, which can be estimated by minimizing the following log-likelihood cost (for i.i.d samples):

$$-\log\left(p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{\phi})\right) = -\sum_{n=1}^{N} \log\left(p(y_n|\theta_1(\boldsymbol{x}_n), \ldots, \theta_J(\boldsymbol{x}_n), \boldsymbol{\phi})\right). \tag{4.3}$$

Remarkably, the deep model in equation (4.2) allows exploiting the representation learning capability of neural networks within a chained framework through the likelihood in equation (4.3).

4.2 Regularized Chained Deep Neural Network for Classification with Multiple Annotators

We follow the model proposed by authors in [27]; here, a Regularized Chained Deep Neural Network–(RCDNN) is introduced for classification tasks from multiple annotations. Concerning this, let $\lambda_n^{(r)} \in \{0, 1\}$ be a binary variable representing the r-th annotator reliability: $\lambda_n^{(r)} = 1$ if $y_n^{(r)} = y_n$, and $\lambda_n^{(r)} = 0$ in other cases. If $\lambda_n^{(r)} = 1$, the label $y_n^{(r)}$ is modeled by means of a categorical distribution; otherwise, if $\lambda_n^{(r)} = 0$, $y_n^{(r)}$ is supposed to follow a uniform distribution. In consequence, the likelihood function in equation (4.3) is rewritten as:

$$p(\boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{r \in R_n} \left(\prod_{k=1}^{K} \zeta_{n,k}^{\delta(y_n^{(r)} - k)} \right)^{\lambda_n^{(r)}} \left(\frac{1}{K} \right)^{1 - \lambda_n^{(r)}}, \tag{4.4}$$

where $\delta(y_n^{(r)} - k) = 1$, if $y_n^{(r)} = k$, and $\delta(y_n^{(r)} - k) = 0$, otherwise. Moreover, $\zeta_{n,k} = p(y_n^{(r)} = k|\lambda_n^{(r)} = 1)$ is the estimation of the hidden ground truth for the n-th instance in class k.

Accordingly, an architecture holding $J = K + R$ outputs is required within our RCDNN for modeling the likelihood parameters $\boldsymbol{\theta}$ in equation (4.4). In particular, K output layers are fixed to estimate the hidden ground truth $\zeta_{n,k}$ based on a softmax function as follows:

$$\zeta_{n,k} = \frac{\exp(f_k(\boldsymbol{x}_n))}{\sum_{i=1}^{K} \exp(f_i(\boldsymbol{x}_n))}. \tag{4.5}$$

Furthermore, a step function can compute the annotator's reliability. Yet, the step function is approximated through R output layers $\{\varsigma_r(\cdot)\}_{r=1}^{R}$, fixing the well-known sigmoid activation to avoid discontinuities and favor the RCDNN implementation:

$$\lambda_n^{(r)} = \varsigma(f_m(\boldsymbol{x}_n)) = \frac{1}{1+\exp(f_{l_r}(\boldsymbol{x}_n))}, \tag{4.6}$$

where $l_r = K + r \in \{K+1, \dots J\}$ is the index of the output linked to the estimation of the reliability of r-th expert. Afterward, the log-likelihood of equation (4.4) is used to compute the RCDNN weights and bias in $\boldsymbol{\phi}$, as follows:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} - \sum_{n=1}^{N} \sum_{r \in R_n} \left[\lambda_n^{(r)}(\boldsymbol{\phi}) \left(\sum_{k=1}^{K} \delta(y_n^{(r)} - k) \log\left(\zeta_{n,k}(\boldsymbol{\phi})\right) \right) - \left(1 - \lambda_n^{(r)}(\boldsymbol{\phi})\right) \log(K) \right], \tag{4.7}$$

where $\lambda_n^{(r)}(\boldsymbol{\phi})$ and $\zeta_{n,k}(\boldsymbol{\phi})$ highlight the dependency between the annotator reliability/ground truth estimation and the RCDNN weights and bias.

In turn, to avoid overfitting and favor the RCDNN generalization capability, l1 and l2 norm-based regularizers are used for dense layers; besides, dropout layers are also added. Both regularization schemes are implemented through the function composition presented in equation (4.2). Lastly, to exploit the RCDNN generalization, the well-known Monte-Carlo dropout prediction strategy is used to estimate the expert's reliability $\hat{\lambda}_n^{(r)}$ and the ground truth label $\hat{\zeta}_{n,k}$, as follows [73]:

$$\hat{\lambda}_n^{(r)} = \frac{1}{E} \sum_{e=1}^{E} \lambda_n^{(r)}(\boldsymbol{\phi}^*, \Delta_e), \tag{4.8}$$

$$\hat{\zeta}_{n,k}^{(r)} = \frac{1}{E} \sum_{e=1}^{E} \zeta_{n,k}^{(r)}(\boldsymbol{\phi}^*, \Delta_e); \tag{4.9}$$

where notation $\lambda_n^{(r)}(\boldsymbol{\phi}^*, \Delta_e)$ and $\zeta_{n,k}^{(r)}(\boldsymbol{\phi}^*, \Delta_e)$ stands for the dependency between the estimated output, the trained RCDNN weights and bias based on equation (4.7), and the set Δ_e holding dropout layers. As seen, the Monte-Carlo dropout-based predictions in equations (4.8) and (4.9) compute the RCDNN outputs as the sample mean over a stack of E predictions; each activates the dropout layers in Δ_e randomly for the u-th iteration within a Monte-Carlo scheme. For RCDNN's implementation details, see Section 4.3.4.

4.3 Experimental Set-Up

4.3.1 Tested Datasets

The introduced RCDNN classifier for multiple annotators scenarios is tested in three different datasets. The first category, termed *2D-PCA iris dataset*, is intended to show our method's work graphically. The Principal Component Analysis-(PCA) algorithm is applied to reduce the well-known Iris dataset dimension from four to two [73], aiming to easily observe some pre-

liminary results in a cartesian plane and illustrate how multiple annotations can be simulated. Second, we build *semi-synthetic data* from eight datasets devoted to the binary and multi-class classification of the well-known UCI repository (see Section 2.3.1). Moreover, we test the publicly available bearing data collected by the Case Western Reserve University–(Western). The aim is to build a system to diagnose an electric motor's status based on two accelerometers. The feature extraction was performed as in [74] ($P = 7$, $N = 3413$, $K = 4$). Finally, our proposal is tested on three *fully real datasets* (Voice, Polarity, and Music), where, the input features and the annotations are captured from real-world problems. The description for the three datasets is found in Section 2.3.1; Table 2.1 summarizes the tested datasets.

4.3.2 Provided and Simulated Annotations

Since the *semi-synthetic* datasets do not provide annotations from multiple labelers, to test our RCDNN classifier, it is necessary to simulate those annotations based on the ground truth, which is available for these kind of experiments. Taking into account that our approach is built under the consideration that the annotators' performance depends on the input space, we use the schemes *Non-homogeneous labels* and *Biased coin (Non-homogeneous)* presented in Section 3.3.1.

Regarding the voice quality dataset, the annotations from four experts are provided, $R = 4$. However, only the G, R, and B scales are studied for concrete testing. Indeed, for scales A and S, the sources' expertise is not satisfactory [21]. Similarly, labels from 203 workers are available for the polarity sentiment dataset. Annotators who labeled at least 15% of the available instances are kept, yielding $R = 7$ labelers. Finally, 2946 labels were obtained from 44 instances concerning the music dataset. Nevertheless, in our experiments, the sources that labeled at least 15% of the available instances are studied ($R = 9$).

4.3.3 Method Comparison and Quality Assessment

Our model's validation is carried out by estimating the classification performance as the Area Under the Curve (AUC) and the overall accuracy (Acc). The AUC is extended for multi-class scenarios, as discussed in [72]. A cross-validation scheme is used with 30 repetitions, where 70% of the samples are utilized for training and the remaining 30% for testing, except for the music and polarity dataset, since they clearly define the training and testing sets. Table 4.1 displays the state-of-the-art models that are considered for comparison purposes. The Matlab codes for the state-of-the-art methods studied are publicly available [1]. We highlight that the GPC-Gold is used only to provide an upper bound for our approach.

[1]GPC-MV MA-LFC, MA-MAE, MA-DGRL, GPC-GTIC, KAAR, and LKAAR codes: https://github.com/juliangilg. MA-GPC codes: http://www.fprodrigues.com/

Table 4.1 A short overview of the tested state-of-the-art approaches

Approach	Brief description
GPC-GOLD	A GPC using the real labels (upper bound).
GPC-MV	A GPC that uses the majority voting of the labels as the ground truth.
MA-LFC [1]	A LRC with constant parameters across the input space.
MA-DGRL [27]	A multi-labeler approach that considers as latent variables the annotator performance.
MA-MAE [50]	A LRC where the source parameters depend on the input space.
MA-GPC [20]	A multi-labeler GPC, which is an extension of MA-LFC using a non-linear approach.
KAAR [54]	A kernel-based approach that employs a convex combination of GPC, it codes the labelers dependencies.
LKAAR-(LR,SVM,GPC) [54]	A localized kernel alignment-based annotator relevance analysis using a combination of LRC, SVM, GPC, respectively. It models the annotators' dependencies and the relationship between the labelers' behavior and the input features.

Note: GPC: Gaussian Processes classifier, LRC: logistic regression classifier, MV: majority voting, MA: multiple annotators, MAE: Modelling annotators expertise, LFC: Learning from crowds, DGRL: Distinguishing good from random labelers, KAAR: kernel alignment-based annotator relevance analysis, LKAAR: localized kernel alignment-based annotator relevance analysis

4.3.4 RCDNN Detailed Architecture and Training

The proposed RCDNN architecture for multiple annotators comprises:

- **IN**: An input layer fed by the input samples $\boldsymbol{X} \in \mathbb{R}^{N \times P}$.
- $\varrho_1(\cdot)$: A dense layer coding relevant patterns from input features to perform. The number of neurons is set as $h = \lfloor \rho P \rceil$, where $\rho \in \{0.5, 1, 1.5\}$ is chosen empirically; a linear-based activation function is used to code input data linear dependencies.
- $\varrho_2(\cdot)$: A dense layer fixing a *tanh*-based activation function with $J = K + R$ neurons to reveal non-linear relationships.
- $\varrho_3(\cdot)$: A fully-connected layer with K neurons and a *softmax*-based activation function, which is employed to estimate the hidden ground truth $\zeta_{k,n}$.
- $\varrho_4(\cdot)$: A dense layer with R neurons and a *sigmoid*-based activation function, which is used to compute the annotators' reliability in $\lambda_n^{(r)}$.
- For all provided ϱ_s layers l1 plus l2-based regularization strategy is used, searching the regularization weights within the range {1e-3,1e-2,1e-1}.
- BatchNormalization and Dropout layers are included between layers to avoid vanishing and exploding gradient issues. Also, it favors the RCDNN's generalization capability, as exposed in Section 4.2. See Figure 4.1 for details.
- The optimization problem in equation (4.7) is solved using a Back-propagation algorithm as usual. Moreover, we utilize a mini-batch-based gradient descent approach with automatic differentiation (RMSprop-based optimizer is fixed) to favor scalability.

We clarify that our RCDNN is flexible and admits different deep structures, such as Recurrent or Convolutional layers aiming to deal with complex tasks (*e.g.*, computer vision or natural language processing). Moreover, our approach can build from different activation functions (RELU, ELU, sigmoid, softmax). However, the last layers (in this case ϵ_3 and ϵ_4) need to be designed

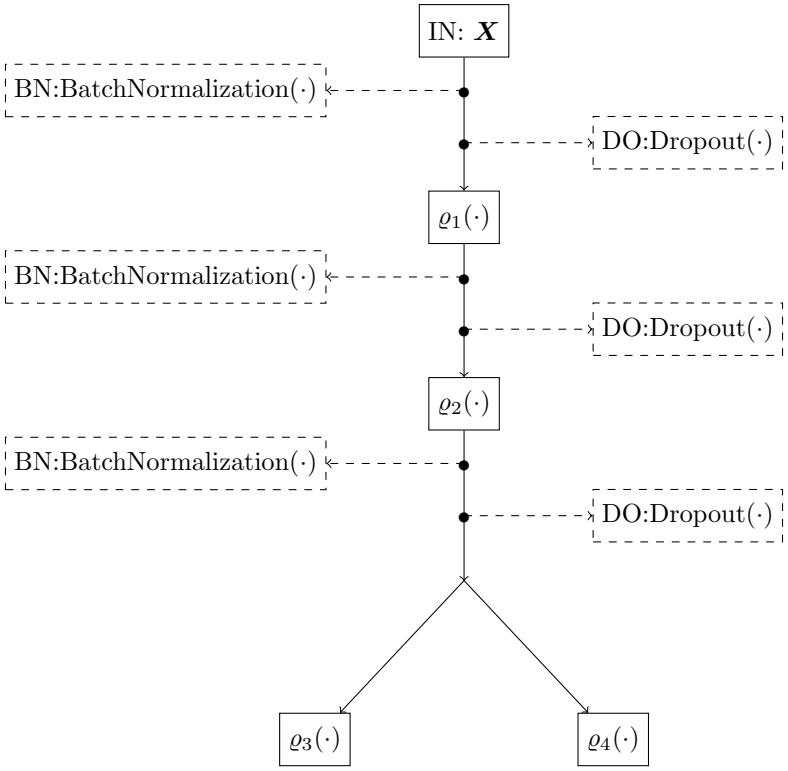

Figure 4.1 RCDNN architecture details. ϱ_s stands for dense layer. ϱ_1 holds a linear activation, ϱ_2 includes a tanh-based activation, and ϱ_3 and ϱ_4 output the hidden ground truth label and the annotator's reliability fixing a softmax and a sigmoid activation, respectively

to code each annotator's behavior and the hidden ground truth. For example, the parameter $\lambda_n^{(r)}$ represents an estimation of the annotators' reliability; accordingly, we need to use an activation function whose output belongs to the range $[0, 1]$.

4.4 Results and Discussion

A controlled experiment is performed to estimate the performance of inconsistent labelers as a function of the input space while highlighting their dependencies. For this first experiment, the 2D PCA Iris dataset is employed (see Section 4.3.1). Besides, the data is divided into five clusters using the K-means technique to emulate five annotators using the approach "Biased coin (Non-homogeneous)". A matrix $\boldsymbol{P} \in [0, 1]^{R \times R}$ is used to set a different score (annotator reliability) for each pair annotator-cluster, as follows:

$$\boldsymbol{P} = \begin{pmatrix} 0 & 0.9 & 0.5 & 0.15 & 0.6 \\ 0.9 & 0 & 0.3 & 0.4 & 0.75 \\ 0.5 & 0.3 & 0 & 0.6 & 0.3 \\ 0.15 & 0.4 & 0.6 & 0 & 0.8 \\ 0.6 & 0.75 & 0.3 & 0.8 & 0 \end{pmatrix}. \tag{4.10}$$

Note that the value $p_{c,r}$ refers to the probability that the annotator r fails to label a sample that belongs to cluster c; thus, a zero value means a perfect annotator for the correspondent cluster. The r-th annotator is an expert (its labels correspond to the ground truth) in the region $c = r$.

Figure 4.2 shows the decision boundaries generated by our approach for the first experiment. As shown, RCDNN offers a suitable representation for the multi-class classification problem; an AUC score of 0.9837 is achieved, which demonstrates its generalization capability, even in cases where the ground truth is unknown. Indeed, RCDNN codes the relationship between the input space, the annotator's behavior and the dependencies among their labels, improving the expert codification quality [10, 54]. To empirically support the above statement, Figure 4.3 shows each annotator's simulated accuracy and the reliability estimated by our RCDNN. The latter elucidates how our method successfully identifies the zones where the labelers have the best accuracy. The above is not unexpected because the annotators' accuracy (simulated) is compared with their reliability (estimated); hence, the regions where a specific labeler obtains the higher accuracy should match the regions where the estimated reliability is closer to 1.

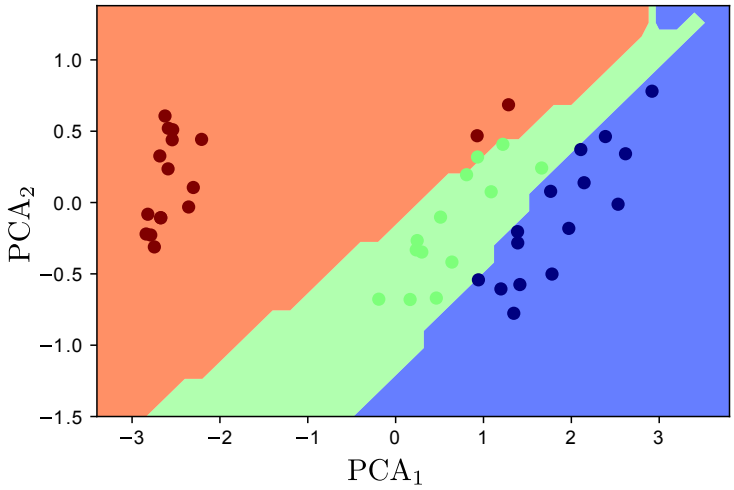

Figure 4.2 RCDNN's decision boundaries for the 2D-PCA Iris dataset (synthetic scenario). AUC= 0.9837. The point's color stands for the Iris dataset classes. PCA$_1$ and PCA$_2$ stand for the first and second PCA-based projections

In addition, Figure 4.4 shows a comparison between the Pearson correlation coefficients (absolute value) from the labelers' performance in equation (4.10), configuring the simulated dependencies among the annotators, and the Pearson correlation coefficients (absolute value) from the weight matrix $\mathbf{\Phi}_{\varrho_4} \in \mathbb{R}^{(K+R)\times R}$ of the layer $\varrho_4(\cdot)$ (RCDNN annotators' dependencies estimation). Comparing the real and the estimated dependencies, it is noticeable that, even though the exact matrix is not recovered, our approach efficiently finds tendencies between annotators' performances. The learned representation from hidden layers (Figure 4.1) allows coding both linear and non-linear patterns that recover the expert dependencies from data. Then, our deep model estimates the unknown ground truth and the relationships between annotators.

Table 4.2(a) shows the results concerning the "Non-homogeneous labels," where it is sup-

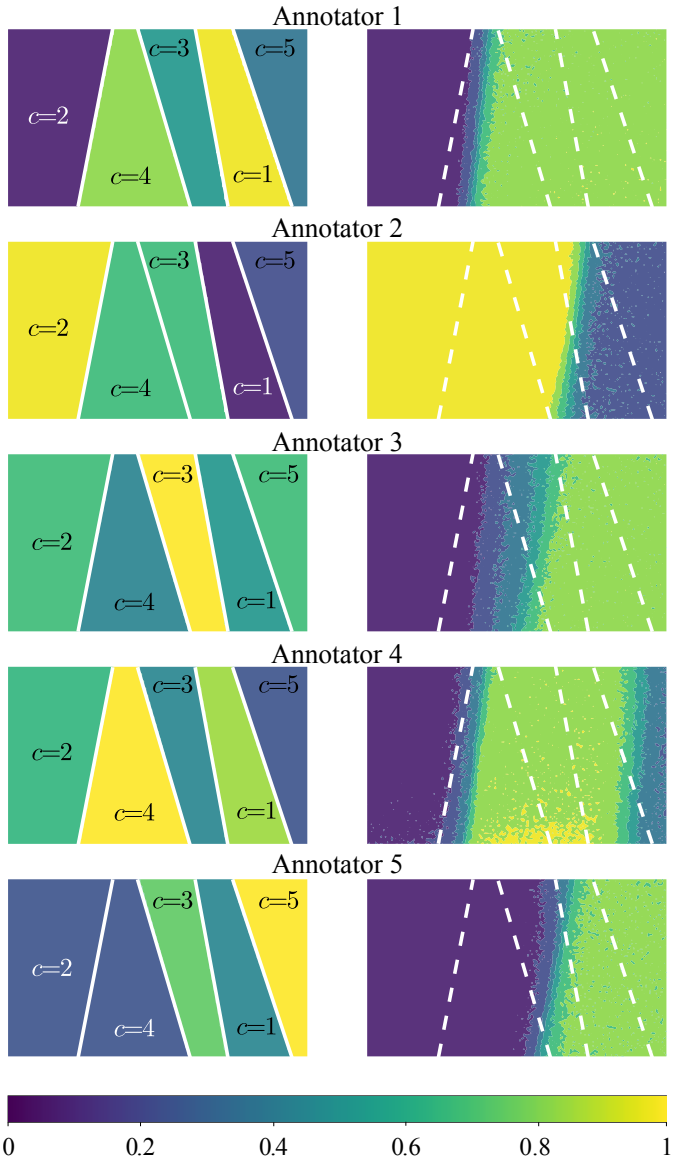

Figure 4.3 RCDNN-based annotators' performance (reliability) estimation for the synthetic experiments (2D PCA Iris data). In the first column (from top to bottom), the simulated accuracy for each annotator is presented based on equation (4.10). The second column shows (from top to bottom) the estimated annotators' reliability (λ_r)

posed that the labelers' performance depends on the input space $\mathcal{X}$. First, we notice that most of the classification schemes present a considerably high performance for both AUC and Acc; in fact, the average AUC and Acc for all methods (except MA-DGRL and MA-MAE) are similar compared to the upper bound GPC-GOLD. The above behavior demonstrates high-quality labels, which is confirmed considering the performance of the most naive approach GPC-MV. Furthermore, our RCDNN presents the best average ranking and the second AUC and Acc scores. Then, from non-linear-based approaches, we notice that a naive approach, such as GPC-MV, obtains similar performance compared with sophisticated ones, like KAAR, LKAAR-SVM, and LKAAR-GPC. Nevertheless, as we already comment, such an outcome is a consequence of simulating annotators with suitable quality, which favors the majority voting method. Besides, MA-

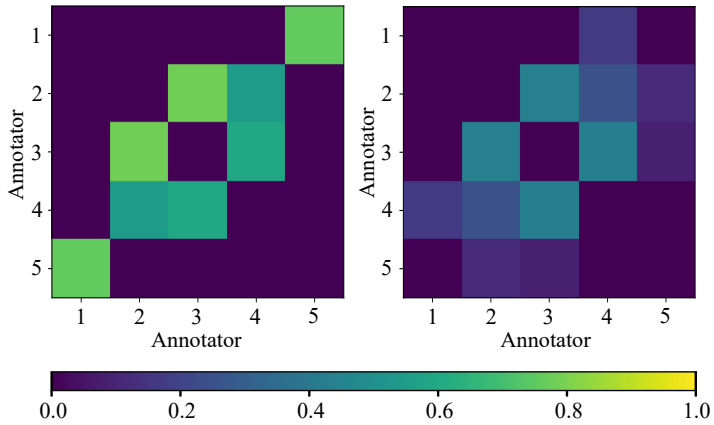

Figure 4.4 Target and estimated annotators' dependencies for the synthetic 2D PCA Iris dataset. On the left, the Pearson correlation coefficients (absolute value) from simulated accuracies (experts reliability) in matrix $\boldsymbol{P}$ of equation (4.10) are shown. On the right, the dependencies among the annotators estimated from the RCDNN ϱ_4 layer's weights are displayed

GPC presents the lowest average ranking compared with its other non-linear methods, resulting from a lack of generalization (overfitting). Regarding the results for the linear models, they achieve lower performance than non-linear ones.

On the other hand, Table 4.2(b) shows the results concerning the simulation method "Biased coin (Non-homogeneous)." At first sight, a generalized lower performance exists compared with previous results in Table 4.2(a). To explain such an outcome, we recall the stimulation parameters $\boldsymbol{P}$ in equation (4.10), where the element $1 - p_{c,r}$ (column r, row c) indicates the r-th annotator's performance in region c. Accordingly, taking the average by column to the matrix $1 - \boldsymbol{P}$, we obtain the annotator's accuracy $[0.57,\ 0.53,\ 0.66,\ 0.61,\ 0.51]$. We remark that the labelers' accuracy is considerably low for this experiment, which impacts the algorithms' performance. RCDNN achieves the best predictive performance in both the overall accuracy and the AUC score; RCDNN obtains the best average ranking. Moreover, the non-linear competitors KAAR, LKAAR-GPC, and LKAAR-SVM achieve competitive results. However, GPC-MV and MA-GPC offer the lowest classification scores. Regarding GPC-MV, the result is explained because GPC-MV corresponds to the most naive approach. After all, it considers that the whole annotators achieve similar performance. On the other hand, the MA-GPC achieves a similar performance compared with GPC-MV; such a behavior proves that MA-GPC is more prone to overfitting [19]. Remarkably, simple classifiers such as MA-LFC, MA-DGRL, and LKAAR-LR obtain competitive outcomes compared to the non-linear competitors; in fact, all the linear models except MA-MAE outperform GPC-MV and MA-GPC. An additional experiment is conducted: an LR-based classifier using the ground truth (following a similar scheme for GPC-GOLD) is trained overall datasets, obtaining an average AUC equal to 87.21 (close enough to the MA-DGRL and LKAAR-LR performances). Accordingly, a linear structure is presented in some of the studied datasets. In turn, MA-MAE obtains the worst generalization performance (even worse than GPC-MV). This outcome is a consequence of overfitting, empirically demonstrated

Table 4.2 Semi-synthetic datasets results

(a) Non-homogeneous labels

Method		breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	western	Average
GPC-GOLD	AUC[%]	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	97.65 ± 2.71	99.22 ± 0.67	90.08 ± 1.94	94.52 ± 0.57	92.39
	Acc[%]	96.44 ± 1.54	68.48 ± 4.43	91.08 ± 2.41	76.71 ± 1.96	99.16 ± 0.85	95.85 ± 3.29	96.92 ± 1.44	70.68 ± 6.81	79.75 ± 0.57	86.12
GPC-MV	AUC[%]	99.11 ± 0.58	70.95 ± 3.90	93.14 ± 3.49	81.21 ± 2.57	87.83 ± 4.11	99.63 ± 0.39	98.41 ± 1.38	91.48 ± 1.48	78.14 ± 4.15	88.87
	Acc[%]	96.29 ± 1.48	66.60 ± 4.31	87.90 ± 3.26	74.87 ± 2.32	81.96 ± 3.46	95.33 ± 3.05	93.96 ± 3.34	82.68 ± 5.30	63.35 ± 1.68	82.54
MA-LFC	AUC[%]	98.72 ± 0.93	**71.53 ± 4.18**	82.08 ± 4.79	82.29 ± 2.22	61.13 ± 3.28	98.75 ± 1.44	96.83 ± 1.75	**99.58 ± 0.11**	87.77 ± 0.79	86.72
	Acc[%]	95.63 ± 1.79	**69.68 ± 4.20**∗	81.43 ± 4.44	**76.52 ± 1.91**∗	64.88 ± 2.86	94.44 ± 4.62	87.74 ± 4.67	**95.40 ± 0.71**∗	57.21 ± 1.32	80.32
MA-DGRL	AUC[%]	99.30 ± 0.39	68.00 ± 4.09	77.60 ± 7.50	81.72 ± 2.57	61.83 ± 2.80	98.78 ± 1.34	95.33 ± 3.35	98.31 ± 0.32	87.67 ± 0.85	85.39
	Acc[%]	94.63 ± 1.77	65.77 ± 3.47	81.94 ± 3.42	76.45 ± 2.81	66.45 ± 2.24	94.59 ± 2.96	84.91 ± 6.50	89.58 ± 0.99	60.55 ± 1.26	79.43
MA-MAE	AUC[%]	99.28 ± 0.60	70.82 ± 3.90	78.91 ± 6.01	81.80 ± 2.57	60.35 ± 3.28	85.97 ± 2.39	98.20 ± 1.33	97.27 ± 0.28	72.83 ± 0.80	82.76
	Acc[%]	96.31 ± 1.38	66.92 ± 3.35	82.25 ± 3.99	76.12 ± 2.77	65.64 ± 2.37	94.81 ± 4.14	89.31 ± 5.79	92.94 ± 0.76	52.41 ± 1.56	79.63
MA-GPC	AUC[%]	95.81 ± 2.94	49.81 ± 11.72	94.46 ± 3.09	67.83 ± 4.24	81.44 ± 3.81	99.15 ± 1.03	99.85 ± 0.24	99.42 ± 0.14	**94.14 ± 0.52**	86.87
	Acc[%]	96.70 ± 1.37∗	59.52 ± 4.71	82.13 ± 3.32	72.77 ± 2.71	76.39 ± 2.85	94.30 ± 2.90	94.34 ± 2.80	94.74 ± 0.68	**78.52 ± 1.11**∗	**83.26**∗
KAAR	AUC[%]	98.81 ± 0.66	70.20 ± 5.70	93.88 ± 3.53	81.18 ± 2.93	89.55 ± 2.84	99.56 ± 0.52	99.53 ± 0.36	92.34 ± 1.38	81.77 ± 1.02	89.64
	Acc[%]	96.02 ± 1.14	65.99 ± 5.44	87.52 ± 4.24	75.10 ± 2.98	81.68 ± 2.41	**95.56 ± 2.92**∗	96.54 ± 2.11	81.11 ± 4.15	64.58 ± 1.47	82.67
LKAAR-LR	AUC[%]	99.34 ± 0.44	68.86 ± 5.16	87.14 ± 3.38	82.04 ± 2.44	65.40 ± 3.13	96.00 ± 2.50	99.21 ± 0.82	97.97 ± 0.27	83.25 ± 1.22	86.57
	Acc[%]	96.00 ± 1.46	64.17 ± 4.22	84.10 ± 3.20	75.67 ± 2.15	66.96 ± 2.74	82.59 ± 6.07	94.28 ± 3.19	90.02 ± 0.93	51.49 ± 2.05	78.36
LKAAR-SVM	AUC[%]	98.29 ± 0.80	64.37 ± 3.36	**96.98 ± 2.01**	77.80 ± 2.28	89.82 ± 2.14	98.05 ± 1.90	99.53 ± 0.47	97.89 ± 0.32	79.08 ± 0.95	89.09
	Acc[%]	96.36 ± 1.02	63.14 ± 3.67	**92.19 ± 2.43**∗	72.52 ± 2.22	80.99 ± 2.81	84.44 ± 6.76	96.48 ± 2.26	91.28 ± 0.93	53.73 ± 2.06	81.79
LKAAR-GPC	AUC[%]	99.00 ± 0.75	71.07 ± 5.05	93.37 ± 2.91	81.23 ± 2.21	**91.97 ± 2.01**	99.57 ± 0.61	99.64 ± 0.34	92.61 ± 1.73	81.37 ± 1.37	**89.98**
	Acc[%]	96.03 ± 1.32	66.92 ± 4.79	87.75 ± 3.90	75.10 ± 2.65	**84.09 ± 2.43**∗	95.26 ± 3.29	96.54 ± 2.16	80.98 ± 3.91	65.20 ± 1.72	82.65
RCDNN (ours)	AUC[%]	**99.47 ± 0.33**	69.80 ± 6.07	92.60 ± 2.80	**83.25 ± 3.13**	71.17 ± 3.76	**99.74 ± 0.26**	**99.90 ± 0.13**	99.15 ± 0.19	89.61 ± 0.71	89.41
	Acc[%]	**97.06 ± 1.19**∗	63.69 ± 4.26	86.79 ± 2.37	76.00 ± 3.10	68.06 ± 3.02	95.33 ± 2.46	**97.84 ± 1.86**∗	92.96 ± 1.06	66.46 ± 1.82	82.68

(b) Biased coin (Non-homogeneous) labels

Method		breast	bupa	ionosphere	pima	tic-tac-toe	iris	wine	segmentation	western	Average
GPC-GOLD	AUC[%]	99.04 ± 0.94	72.21 ± 3.69	95.02 ± 2.44	83.76 ± 1.98	99.97 ± 0.06	97.65 ± 2.71	99.22 ± 0.67	90.08 ± 1.94	94.52 ± 0.57	92.39
	Acc[%]	96.44 ± 1.54	68.48 ± 4.43	91.08 ± 2.41	76.71 ± 1.96	99.16 ± 0.85	95.85 ± 3.29	96.92 ± 1.44	70.68 ± 6.81	79.75 ± 1.28	86.12
GPC-MV	AUC[%]	90.78 ± 4.28	50.47 ± 6.19	82.91 ± 6.03	70.18 ± 6.29	65.91 ± 6.72	98.55 ± 1.38	97.75 ± 2.04	90.18 ± 1.71	74.40 ± 4.94	80.13
	Acc[%]	86.63 ± 2.06	48.27 ± 4.84	75.65 ± 6.45	66.52 ± 5.16	64.66 ± 3.64	88.81 ± 5.00	86.92 ± 5.76	79.24 ± 4.99	65.04 ± 1.52	73.53
MA-LFC	AUC[%]	97.99 ± 0.99	59.64 ± 8.08	72.66 ± 9.98	72.73 ± 3.43	52.88 ± 3.13	96.72 ± 8.98	96.47 ± 2.13	**99.50 ± 0.15**	84.97 ± 0.84	81.51
	Acc[%]	**96.00 ± 1.70**∗	56.41 ± 8.12	69.17 ± 12.53	58.10 ± 4.53	46.27 ± 3.03	92.30 ± 5.18	87.55 ± 4.97	**95.06 ± 0.80**∗	55.17 ± 1.33	72.89
MA-DGRL	AUC[%]	99.31 ± 0.42	61.77 ± 6.17	77.83 ± 7.02	81.66 ± 2.65	55.70 ± 3.95	98.76 ± 1.33	95.26 ± 3.30	98.32 ± 0.34	86.61 ± 1.10	83.91
	Acc[%]	78.08 ± 2.22	55.64 ± 4.52	71.43 ± 5.15	**76.90 ± 1.99**∗	60.64 ± 2.33	94.37 ± 2.66	84.84 ± 6.32	89.63 ± 0.89	65.61 ± 1.28	75.24
MA-MAE	AUC[%]	95.22 ± 1.70	64.63 ± 9.77	64.18 ± 9.17	79.94 ± 2.64	52.36 ± 4.78	93.16 ± 5.08	96.25 ± 2.40	94.40 ± 1.26	61.40 ± 0.93	77.95
	Acc[%]	87.15 ± 1.85	62.34 ± 8.46	67.94 ± 7.19	75.94 ± 2.69	53.33 ± 6.42	81.70 ± 11.68	86.67 ± 5.15	88.38 ± 2.00	49.34 ± 4.15	72.53
MA-GPC	AUC[%]	85.37 ± 5.90	40.79 ± 12.30	74.52 ± 4.57	73.17 ± 3.34	61.82 ± 4.51	98.71 ± 1.14	99.60 ± 0.41	99.35 ± 0.14	**93.09 ± 0.58**	80.71
	Acc[%]	92.55 ± 2.17	52.82 ± 6.38	69.87 ± 4.41	62.42 ± 3.00	62.33 ± 2.98	93.85 ± 3.49	95.09 ± 2.65	93.46 ± 0.83	**76.88 ± 1.19**∗	77.70
KAAR	AUC[%]	97.81 ± 0.99	56.52 ± 9.13	82.20 ± 4.93	67.90 ± 3.16	75.34 ± 4.70	98.75 ± 1.10	97.91 ± 1.36	91.75 ± 1.41	82.30 ± 0.73	83.39
	Acc[%]	77.19 ± 3.14	52.44 ± 7.79	72.60 ± 4.80	61.20 ± 2.95	70.69 ± 3.63	90.44 ± 5.48	91.45 ± 4.28	76.38 ± 5.05	64.61 ± 1.36	73.00
LKAAR-LR	AUC[%]	**99.52 ± 0.30**	**66.07 ± 6.14**	82.99 ± 5.01	80.57 ± 3.31	52.32 ± 3.38	96.83 ± 2.14	99.27 ± 0.68	97.87 ± 0.30	81.03 ± 0.80	84.05
	Acc[%]	92.47 ± 2.24	**60.22 ± 5.67**∗	78.92 ± 4.32	75.07 ± 2.65	55.64 ± 2.77	83.41 ± 6.92	94.59 ± 3.12	89.77 ± 0.99	54.80 ± 2.05	76.10
LKAAR-SVM	AUC[%]	98.37 ± 1.00	52.35 ± 6.40	**88.28 ± 5.13**	66.84 ± 3.66	73.85 ± 3.43	96.22 ± 2.50	98.88 ± 0.80	97.59 ± 0.34	79.19 ± 1.46	82.39
	Acc[%]	87.72 ± 5.17	50.96 ± 6.81	**84.73 ± 4.66**∗	64.81 ± 3.11	70.02 ± 2.74	74.15 ± 7.90	91.82 ± 4.33	90.37 ± 1.24	55.39 ± 3.03	74.44
LKAAR-GPC	AUC[%]	98.14 ± 1.04	58.36 ± 7.24	86.23 ± 4.47	73.80 ± 2.83	**80.02 ± 4.15**	**99.61 ± 0.61**	98.74 ± 0.93	92.24 ± 1.80	83.35 ± 0.75	85.61
	Acc[%]	86.76 ± 4.33	54.52 ± 5.27	78.25 ± 5.51	69.64 ± 3.01	**74.90 ± 2.99**∗	**95.93 ± 3.15**∗	93.84 ± 3.57	78.71 ± 4.18	66.58 ± 1.19	77.68
RCDNN (ours)	AUC[%]	99.26 ± 0.42	64.16 ± 3.87	83.41 ± 6.28	**82.08 ± 3.27**	65.31 ± 3.87	99.51 ± 0.53	**99.77 ± 0.22**	99.06 ± 0.20	87.94 ± 1.03	**86.72**
	Acc[%]	94.07 ± 2.00	58.24 ± 5.13	76.70 ± 6.19	74.91 ± 3.77	65.07 ± 1.17	93.33 ± 3.30	**96.17 ± 2.57**∗	91.28 ± 0.99	61.56 ± 5.13	**79.04**∗

Note: the highest AUC excluding the upper bound (target) classifier GPC-GOLD. Marked with ∗: the highest accuracy (Acc) except the upper bound. The last column presents the average ranking for the AUC score and the overall accuracy (GPC-GOLD is not considered). The best average ranking for AUC is highlighted in bold, and the accuracy is marked with ∗

in [19]. It is noteworthy that RCDNN and LKAAR-GP obtain similar results, which is expected since both approaches compute the annotators' performance as a function of the input space while considering dependencies between the labelers. However, an unexpected result regarding the "tic-tac-toe" dataset arises, where LKAAR-GP far exceeds our approach's performance. The categorical features cause the above outcome in such a dataset, which cannot be modeled with the chosen DNN architecture Figure 4.1. Still, our method can be easily adapted by setting different layers and activation functions. It is worth noting that the previous experiments were done under controlled scenarios using simulated annotations to stress our method and compare its performance with recently developed approaches. In short, RCDNN offers the best advantages among the state-of-the-art models considered in AUC, overall accuracy, and average ranking.

Up to this point, RCDNN unravels the information hidden in noisy annotations (simulated) to estimate the unknown ground truth considering experts' performance as a function of the input

Table 4.3 Fully real-world datasets results

AUC([%])						
Method	**Voice Dataset**			**Polarity Dataset**	**Music**	**Average**
	G	**R**	**B**			
GPC-GOLD	93.66	93.66	93.66	80.26	92.84	90.81
GPC-MV	90.17	84.73	84.04	71.14	88.79	83.77
MA-LFC	89.99	90.59	87.27	72.06	85.99	85.18
MA-DGRL	85.45	90.14	79.33	56.13	88.32	79.86
MA-MAE	91.08	89.12	80.74	48.73	81.92	78.31
MA-GPC	91.50	91.16	80.81	61.18	82.53	81.43
KAAR	89.85	93.50	89.20	**77.46**	88.96	87.79
LKAAR-LR	90.39	92.92	88.94	68.28	84.43	84.99
LKAAR-SVM	92.06	93.02	86, 98	72.70	89.98	87.70
LKAAR-GPC	90.78	93.60	89.79	76.50	86.44	87.42
RCDNN	**92.24**	**94.19**	**92.57**	76.04	**93.29**	**89.66**

Note: the method with the highest performance excluding the upper bound (target) classifier GPC-GOLD.

space and dependencies among labelers. However, the following experiments aim to demonstrate how our approach can outperform state-of-the-art methods even for real labelers, *e.g.*, the challenge is higher as the input data and the annotations are obtained from real-world applications. Table 4.3 describes the results achieved using AUC as the metric to compare the state-of-the-art methods in five different real-world datasets.

First, analyzing the voice data for the scales **G** and **R**, all the approaches give similar AUC values. In fact, for scale **G**, the GPC-MV attains competitive performance. The latter can be explained in that the annotators exhibit similar conduct for these scales [21]. Conversely, for the **B** scale, a generalized reduction is presented. Looking at RCDNN results for this database, it is noticeable that the achievement is similar among all the scales, which is an exceptional outcome showing our method's ability to detect regions where annotators have superior execution.

In Polarity Dataset, an acceptable RCDNN's performance is attained compared to others. Our approach requires defining several layer weights in the deep model (Figure 4.1) concerning the number of features (P), labelers (R), and classes (K). For this particular dataset, those values are considerably higher: $P = 1200$, $R = 7$, and $K = 2$. Nevertheless, the introduced regularization strategy (l1, l2, plus Monte-Carlo Dropout) allows computing an acceptable AUC performance of 76.04 in comparison with the best achieved by the KAAR method 77.46.

Lastly, in the case of Music data, our RCDNN obtains the best classification performance. On the other hand, MA-MAE and MA-GPC exhibit a significantly low performance, even lower than the intuitive lower bound (GPC-MV). This behavior has been repeated in previous experiments because of the over-fitting issue. Nevertheless, an additional challenge is presented for the music dataset regarding the multi-class classification setting. Accordingly, a *one-vs-all* scheme is fixed for all binary classification methods (including MA-MAE and MA-GPC). Such a scheme to deal with multi-class classification can lead regions on the input space that are ambiguously classified [9].

As a final experiment, we analyze the impact of spammers and malicious annotators on the

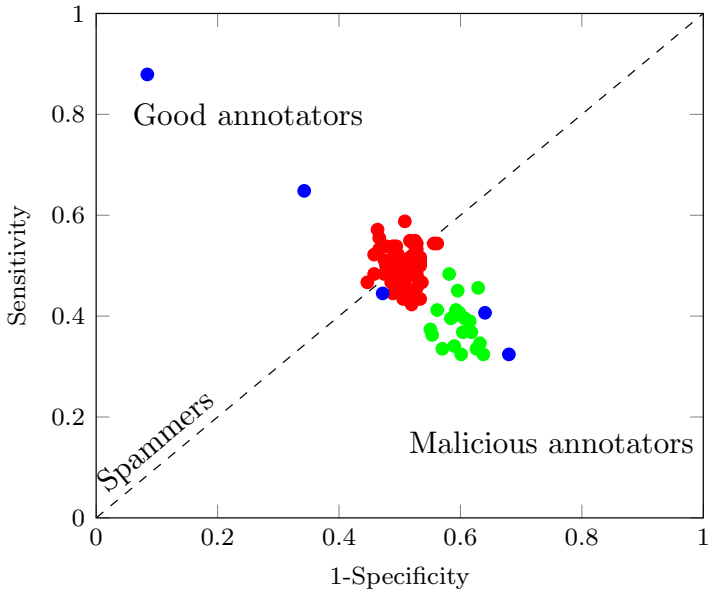

Figure 4.5 Receiver Operating Characteristic (ROC) plot for the annotators simulated within the spammers and malicious scenario. Blue dots indicate the basis annotators. Red dots show extra annotators with parameters $R_e = 65$ and $p_r = 0.5$. Green dots specify extra labelers with $R_e = 20$ and $p_r = 0.6$. We notice that annotators located in the dashed line vicinity are considered Spammers. Similarly, labelers above in the dashed line are regarded as good annotators; conversely, labelers located below such a line are malicious annotators

performance of our multi-labeler classifier. We use the Pima dataset for concrete testing, which holds 768 instances; from this dataset, we use 538 samples for training and the remaining 230 for testing. We create synthetic labels from 5 annotators generated from the Biased coin (non-homogeneous) procedure (Section 4.3.2 and equation (4.10)). According to Figure 4.5 (blue dots), from the 5 labelers, two are categorized as suitable labelers, one as Spammers, and the remaining as Malicious. Then, we add additional R_e annotators aiming to test our approach in extreme scenarios where the number of malicious or spammers annotators increases. The labels are simulated as follows: a random number $\tau_n^{(r)}$ is sampled from a Bernoulli distribution with parameter p_r; then if $\tau_n^{(r)} = 0$, $y_n^{(r)} = y_n$, and $y_n^{(r)} = \tilde{y}_n$ otherwise. For Spammers, we use $R_e = 65$ and $p_r = 0.5$ (see red dots in Figure 4.5); like for malicious labelers, we fix $R_e = 20$ and $p_r = 0.6$ (see green dots in Figure 4.5).

Figure 4.6 presents the classifiers' performance as a function of the number of spammers (left in Figure 4.6) and malicious annotators (Figure 4.6). First, we analyze the effect of Spammers annotators on the RCDNN's performance. From the results in Figure 4.6 (left), we remark that when the number of Spammers is less than 40, the performance of our approach is not affected. However, when the number of Spammers exceeds 40, the RCDNN'S AUC becomes unstable, oscillating between 0.6 and 0.8. Accordingly, we highlight that the critical point is presented when the percentages of good, spammers, and malicious labelers are respectively 4.65%, 90.70%, and 4.65%, which shows that our RCDNN is robust in the presence of a high number of Spammers. Now, we compare our RCDNN with two state-of-the-art models, MA-LFC (linear

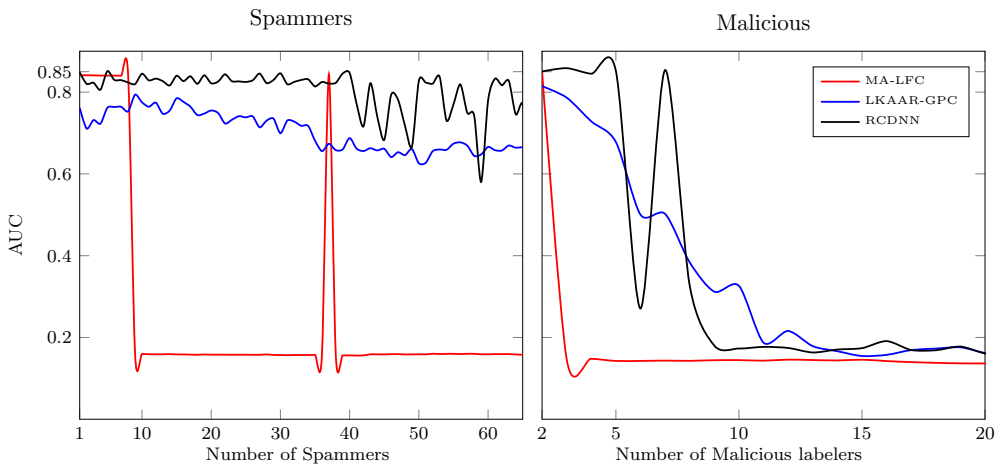

Figure 4.6 MA-LFC, LKAAR-GPC, and RCDNN performance (AUC) as a function of the number of labelers (spammers and malicious annotators)

model with the more competitive performance according to Table 4.3) and LKAAR-GPC (Non-linear model with the more competitive AUC in Table 4.3). We notice that the LKAAR-GPC behavior is similar to our approach; when the number of spammers exceeds 35, the AUC starts to descend gradually. Conversely, we note that the MA-LFC's performance is drastically affected by the spammers. For more than 8 spammers, the AUC is close to 0.2.

Second, we inspect the consequences when malicious labelers are added. From the results in Figure 4.6 (right), we note that our RCDNN is significantly affected when we have more than 5 malicious annotators; in that case, the AUC decreases from 0.85 approximately to a value near 0.2. Thereby, we notice that the critical point is presented when the percentages of good, spammers and malicious labelers are respectively 25%, 12.5%, 62.5%. In such a sense, for this experiment, we can affirm that our approach can deal with malicious labelers if the percentage of them is below 62.5%. Finally, studying the results related to LKAAR-GPC, we notice that LKAAR again performs similarly to our RCDNN due to more than 5 malicious labelers; LKAAR-GPC achieves AUC scores lower than 0.5; on the other hand, MA-LFC susceptible since, for more than 2 malicious labelers, the AUC decreases to a value near to 0.2.

4.5 Summary

This paper introduces a novel Regularized Chained Deep Neural Network classifier, termed RCDNN, to deal with multiple annotator scenarios. Our method is built based on the ideas of the chained Gaussian Processes [59], where each parameter in a multi-labeler likelihood is modeled by using the outputs of a deep neural network. In such a way, RCDNN codes the annotators' expertise as a function of the input data and the dependencies among the labelers from the last hidden layer's weights. Besides, l1, l2, and Monte-Carlo Dropout regularization strategies are coupled within our RCDNN architecture and predictor to contract the overfitting challenge of deep

models. The proposal is tested using different scenarios concerning the provided annotations: synthetic, semi-synthetic, and real-world experts. According to the results, RCDNN achieves robust predictive properties for the studied datasets, outperforming state-of-the-art methods while providing an estimation of each labeler's reliability and the dependencies among annotators.

However, we notice that RCDNN is based on a frequentist approach; thus, the estimations for the annotators' parameters and the supervised learning framework are deterministic, and it is not possible to capture the uncertainty related to them.

Chapter 5 Correlated Chained Gaussian Processes for Multiple Annotators

In this chapter, we propose a probabilistic model, named the correlated chained Gaussian Processes for multiple annotators–(CCGPMA), to jointly build a prediction algorithm applicable to classification and regression tasks. CCGPMA is based on the chained GPs model–(CGP) [59], which is a Multi-GPs framework where the parameters of an arbitrary likelihood function are modeled with multiple independent GPs (one GP prior per parameter). Unlike CGP, we consider that multiple correlated GPs model the likelihood's parameters. For doing so, we take as a basis the ideas from a Multi-output GP–(MOGP) regression [40], where each output is coded as a weighted sum of shared latent functions via a semi-parametric latent factor model–(SLFM) [61]. In contrast to the MOGP, we do not have multiple outputs but multiple functions chained to the given likelihood parameters. From the multiple annotators' point of view, the likelihood parameters are related to the labelers' behavior; CCGPMA models the labelers' behavior as a function of the input features while also considering annotators' interdependencies. Moreover, our proposal is based on the so-called inducing variables framework [75], in combination with stochastic variational inference [76]. To the best of our knowledge, this is the first attempt to build a probabilistic approach to model the labelers' behavior as a function of the input features while also considering annotators' interdependencies. Using both simulated and real-world data, the results show how our method can deal with regression and classification problems from multi-labelers data.

5.1 Chained Gaussian Processes

Let us consider an input-output dataset $\mathcal{D} = \{\boldsymbol{X} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}\}$, where $\boldsymbol{X} = \left\{\boldsymbol{x}_n \in \mathcal{X} \subseteq \mathbb{R}^P\right\}_{n=1}^N$ and $\boldsymbol{y} = \{y_n \in \mathcal{Y}\}_{n=1}^N$. In turn, let a GP be a collection of random variables $f(\boldsymbol{x})$ indexed by the input samples $\boldsymbol{x} \in \mathcal{X}$ holding a joint multivariate Gaussian distribution [37]. A GP is defined by its mean $m(\boldsymbol{x}) = \mathbb{E}\, f(\boldsymbol{x})$ (we consider $m(\boldsymbol{x}) = 0$) and covariance function

$$\kappa_f(\boldsymbol{x}, \boldsymbol{x'}) = \mathbb{E}\,(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'})), \tag{5.1}$$

where $\kappa_f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a given kernel function and $\boldsymbol{x}' \in \mathcal{X}$, yielding:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(0, \kappa_f(\boldsymbol{x}, \boldsymbol{x'})). \tag{5.2}$$

If we consider the finite set of inputs in $\boldsymbol{X}$, then $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ is drawn for a multivariate Gaussian distribution $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{ff}})$, where $\boldsymbol{K}_{\boldsymbol{ff}} \in \mathbb{R}^{N \times N}$ is the covariance

matrix formed by the evaluation of $\kappa_f(\cdot,\cdot)$ over the input set $\boldsymbol{X}$.

Accordingly, using GPs for modeling the input-output data collection $\mathcal{D}$ consists of constructing a joint distribution between a given likelihood function and one or multiple GP-based priors. To code each likelihood parameter as a random process, we employ the so-called chained GP–(CGP) that attaches such parameters to multiple independent GP priors, as follows [59]:

$$p(\boldsymbol{y},\hat{\boldsymbol{f}}|\boldsymbol{X}) = \prod_{n=1}^{N} p(y_n|\theta_1(\boldsymbol{x}_n),\ldots,\theta_J(\boldsymbol{x}_n)) \times \prod_{j=1}^{J} \mathcal{N}(\boldsymbol{f}_j|\boldsymbol{0},\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j}), \tag{5.3}$$

where each $\{\theta_j(\boldsymbol{x}) \in \mathcal{M}_j\}_{j=1}^{J}$ represents the likelihood's parameters, being $J \in \mathbb{N}$ the number of parameters to represent the likelihood. Besides, each $\theta_j(\boldsymbol{x})$ holds a non-linear mapping from a GP prior, *e.g.*, $\theta_j(\boldsymbol{x}) = h_j(f_j(\boldsymbol{x}))$, where $h_j : \mathbb{R} \to \mathcal{M}_j$ is a deterministic function that maps each latent function–(LF) $f_j(\boldsymbol{x})$, to the appropriate domain $\mathcal{M}_j$.

Moreover, $\boldsymbol{f}_j = [f_j(\boldsymbol{x}_1),\ldots,f_j(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ is a LF vector that follows a Gaussian Process prior, and $\hat{\boldsymbol{f}} = [\boldsymbol{f}_1,\ldots,\boldsymbol{f}_J]^\top \in \mathbb{R}^{NJ}$. $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} \in \mathbb{R}^{N\times N}$ is the covariance matrix belonging to the j-th GP prior, which is computed based on the kernel function $\kappa_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The non-parametric formulation of a GP introduces computational loads through the inference process. For instance, considering that the dataset $\mathcal{D}$ configures a regression problem, a GP modeling involves a computational complexity of $\mathcal{O}(N^3)$ to invert the matrix $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j}$ [37]. A common approach to reduce such computational complexity is to augment the GP prior with a set of $M << N$ inducing variables [77] $\boldsymbol{u}_j = [f_j(\boldsymbol{z}_1^{(j)}),\ldots,f_j(\boldsymbol{z}_M^{(j)})]^\top \in \mathbb{R}^M$ through additional evaluations of $f_j(\cdot)$ at unknown locations $\boldsymbol{Z}_j = [\boldsymbol{z}_1^{(j)},\ldots,\boldsymbol{z}_M^{(j)}] \in \mathbb{R}^{M\times P}$, which decreases the GP's computational complexity to $\mathcal{O}(NM^2)$. Further, the following augmented GP prior arises:

$$p(\boldsymbol{f}_j,\boldsymbol{u}_j) = \mathcal{N}\left(\begin{bmatrix}\boldsymbol{f}_j\\ \boldsymbol{u}_j\end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix}\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} & \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_j}\\ \boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{f}_j} & \boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{u}_j}\end{bmatrix}\right), \tag{5.4}$$

where $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_j} \in \mathbb{R}^{N\times M}$ is the cross-covariance matrix formed by the evaluation of the kernel function $\kappa_j(\cdot,\cdot)$ between $\boldsymbol{X}$ and $\boldsymbol{Z}_j$. Likewise, $\boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{u}_j} \in \mathbb{R}^{M\times M}$ is the inducing points-based covariance matrix. Then, the distribution of $\boldsymbol{f}_j$ conditioned to the inducing points $\boldsymbol{u}_j$ can be written as:

$$p(\boldsymbol{f}_j|\boldsymbol{u}_j) = \mathcal{N}\left(\boldsymbol{f}_j|\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_j}\boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{u}_j}^{-1}\boldsymbol{u}_j, \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} - \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_j}\boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{u}_j}^{-1}\boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{f}_j}\right), \tag{5.5}$$

$$p(\boldsymbol{u}_j) = \mathcal{N}\left(\boldsymbol{u}_j|\boldsymbol{0},\boldsymbol{K}_{\boldsymbol{u}_j\boldsymbol{u}_j}\right). \tag{5.6}$$

In most cases equations (5.5) and (5.6) are non-conjugate to the likelihood, finding the posterior distribution $p(\boldsymbol{f},\mathbf{u}|\mathbf{y})$ is not tractable analytically; therefore, we resort to a deterministic approximation of the posterior distribution using variational inference. Hence, the actual posterior can be approximated by a parametrized variational distribution $p(\hat{\boldsymbol{f}},\boldsymbol{u}|\boldsymbol{y}) \approx q(\hat{\boldsymbol{f}},\boldsymbol{u})$ [78], as:

$$q(\boldsymbol{f},\boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u}) = \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}_j)q(\boldsymbol{u}_j), \tag{5.7}$$

where $\boldsymbol{u} = \left[\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_J^\top\right]^\top \in \mathbb{R}^{MJ}$; moreover, $p(\boldsymbol{f}_j|\boldsymbol{u}_j)$ is defined in equation (5.5), and $q(\boldsymbol{u})$ is the posterior approximation over the inducing variables:

$$q(\boldsymbol{u}) = \prod_{j=1}^{J} q(\boldsymbol{u}_j) = \prod_{j=1}^{J} \mathcal{N}\left(\boldsymbol{u}_j|\boldsymbol{m}_j, \boldsymbol{V}_j\right). \tag{5.8}$$

The approximation for the posterior distribution comprises the estimation of the following variational parameters: the mean vectors $\boldsymbol{m}_j \in \mathbb{R}^M$ and the covariance matrices $\boldsymbol{V}_j \in \mathbb{R}^{M\times M}$. Such an assessment is carried out by maximizing an evidence lower bound–(ELBO). Due to space restrictions, the ELBO derivation and details are included in the supplementary material.

5.1.1 Correlated Chained Gaussian Processes

From Section 5.1, we note that the CGP model assumes independence between priors, thereby lacking a correlation structure between GPs. As mentioned before, we consider that the annotators are correlated. We will enable this aspect of the model by assuming dependencies among the latent parameters of the chained GP. In particular, we introduce the correlated chained GPs–(CCGP) to model correlations between the GP latent functions, which are supposed to be generated from a semi-parametric latent factor model–(SLFM) [61]:

$$f_j(\boldsymbol{x}_n) = \sum_{q=1}^{Q} w_{j,q}\mu_q(\boldsymbol{x}_n), \tag{5.9}$$

where $f_j : \mathcal{X} \to \mathbb{R}$ is an LF, $\mu_q(\cdot) \sim \mathcal{GP}(0, k_q(\cdot,\cdot))$ with $k_q : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ being a kernel function, and $w_{j,q} \in \mathbb{R}$ is a combination coefficient ($Q \in \mathbb{N}$). Here, each LF is chained to the likelihood's parameters to extend the joint distribution in equation (5.3) as follows:

$$p(\boldsymbol{y}, \hat{\boldsymbol{f}}, \boldsymbol{u}|\boldsymbol{X}) = p(\boldsymbol{y}|\boldsymbol{\theta}) \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u})p(\boldsymbol{u}), \tag{5.10}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J]^\top \in \mathbb{R}^{NJ}$ holds the model's parameters and $\boldsymbol{\theta}_j = [\theta_j(\boldsymbol{x}_1), \ldots, \theta_j(\boldsymbol{x}_N)]^\top \in \mathbb{R}^N$ relates the j-th parameter with the input space. Our CCGP employs the inducing variables-based method for sparse approximations of GPs [77]. For each $\mu_q(\cdot)$, we introduce a set of $M \leq N$ "pseudo variables" $\boldsymbol{u}_q = [\mu_q(\boldsymbol{z}_1^{(q)}), \ldots, \mu_q(\boldsymbol{z}_M^{(q)})]^\top \in \mathbb{R}^M$ through evaluations of $\mu_q(\cdot)$ at unknown locations $\boldsymbol{Z}_q = [\boldsymbol{z}_1^{(q)}, \ldots, \boldsymbol{z}_M^{(q)}] \in \mathbb{R}^{M\times P}$. Note that $\boldsymbol{u} = \left[\boldsymbol{u}_1^\top, \ldots, \boldsymbol{u}_Q^\top\right]^\top \in \mathbb{R}^{QM}$, yielding:

$$p(\boldsymbol{f}_j|\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{f}_j|\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{u}, \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} - \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{f}_j}\right), \tag{5.11}$$

$$p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}) = \prod_{q=1}^{Q}\mathcal{N}(\boldsymbol{u}_q|\boldsymbol{0}, \boldsymbol{K}_{\boldsymbol{u}_q\boldsymbol{u}_q}), \tag{5.12}$$

where $\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}} \in \mathbb{R}^{QM\times QM}$ is a block-diagonal matrix with blocks $\boldsymbol{K}_{\boldsymbol{u}_q\boldsymbol{u}_q} \in \mathbb{R}^{M\times M}$, based on the kernel function $\kappa_q(\cdot,\cdot)$. On the other hand, the covariance matrix $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} \in \mathbb{R}^{N\times N}$ holds elements $\sum_{q=1}^{Q} w_{j,q}w_{j,q}\kappa_q(\boldsymbol{x}_n, \boldsymbol{x}_{n'})$, with $\boldsymbol{x}_n, \boldsymbol{x}_{n'} \in \boldsymbol{X}$, and $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}} = [\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_1}, \ldots, \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_Q}] \in \mathbb{R}^{N\times QM}$, where $\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}_q} \in \mathbb{R}^{N\times M}$ gathers elements $w_{j,q}\kappa_q(\boldsymbol{x}_n, \boldsymbol{z}_m^{(q)})$, $m \in \{1,\ldots,M\}$. Alike CGP, in most cases, the CCGP posterior distribution $p(\hat{\boldsymbol{f}}, \boldsymbol{u}|\boldsymbol{y})$ has not an analytical solution, so the actual posterior can be approximated by a parametrized variational distribution $p(\hat{\boldsymbol{f}}, \boldsymbol{u}|\boldsymbol{y}) \approx q(\hat{\boldsymbol{f}}, \boldsymbol{u})$, as:

$$q(\hat{\boldsymbol{f}}, \boldsymbol{u}) = p(\hat{\boldsymbol{f}}|\boldsymbol{u})q(\boldsymbol{u}) = \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) \prod_{q=1}^{Q} q(\boldsymbol{u}_q), \tag{5.13}$$

where $p(\boldsymbol{f}_j|\boldsymbol{u})$ is given by equation (5.11), $q(\boldsymbol{u}_q) = \mathcal{N}(\boldsymbol{u}_q|\boldsymbol{m}_q, \boldsymbol{V}_q)$, and $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{V})$. Also, $\boldsymbol{m}_q \in \mathbb{R}^M$, and $\boldsymbol{V}_q \in \mathbb{R}^{M\times M}$ are respectively the mean and covariance of variational distribution $q(\boldsymbol{u}_q)$; similarly, $\boldsymbol{m} = [\boldsymbol{m}_1^\top, \ldots, \boldsymbol{m}_Q^\top]^\top \in \mathbb{R}^{QM}$, and $\boldsymbol{V} \in \mathbb{R}^{QM\times QM}$ is a block-diagonal matrix with blocks given by the covariance matrices $\boldsymbol{V}_q$. We remark that the variational approximation given by equation (5.13) is well known and has been used in several GPs models, including [59, 79]. The approximation for the posterior distribution comprises the computation of the following variational parameters: the mean vectors $\{\boldsymbol{m}_q\}_{q=1}^{Q}$ and the covariance matrices $\{\boldsymbol{V}_q\}_{q=1}^{Q}$. Such an estimation is carried out by maximizing an evidence lower bound–(ELBO), which is given as:

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{f}_1),\ldots,q(\boldsymbol{f}_J)}\left[\log p(y_n|\theta_{1,n},\ldots,\theta_{J,n}\right] - \sum_{q=1}^{Q}\mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q)), \tag{5.14}$$

where $\theta_{j,n} = \theta_j(\boldsymbol{x}_n)$, with $j \in \{1,\ldots,J\}$, and $\mathbb{D}_{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence and $q(\boldsymbol{f}_j)$ is defined as follows:

$$q(\boldsymbol{f}_j) = \mathcal{N}(\boldsymbol{f}_j|\boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{m}, \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{f}_j} + \boldsymbol{K}_{\boldsymbol{f}_j\boldsymbol{u}}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1}(\boldsymbol{V} - \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}})\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\boldsymbol{K}_{\boldsymbol{u}\boldsymbol{f}_j}). \tag{5.15}$$

Yet, in presence of non-Gaussian likelihoods, the variational expectations–(VEs) computation in equation (5.14) cannot be solved analytically [59, 79]. Hence, aiming to model different data types, *i.e.*, classification and regression tasks, we need to find a generic alternative to solve the integrals related to these expectations. In that sense, we use the Gaussian-Hermite quadratures approach as in [59, 77]. We remark such ELBO is used to infer the model's hyperparameters such as the inducing points, the kernel hyperparameters, and the combination factors $w_{j,q}$ equation (5.9). It is worth mentioning that the CCGPs objective functions exhibit an ELBO that allows Stochastic Variational Inference–(SVI) [80]. Hence, the optimization is solved through a *mini-*

batch-based approach from noisy estimates of the global objective gradient, which allows dealing with large scale datasets [59, 77, 79]. Finally, we notice that the computational complexity for our CCGP is dominated by the inversion of $\boldsymbol{K_{uu}}$ with $\mathcal{O}(QM^3)$ and $\boldsymbol{K_{fu}}$ with $\mathcal{O}(JNQM^2)$.

5.2 Correlated Chained GP for Multiple Annotators-CCGPMA

5.2.1 Classification

To model categorical data from multiple annotators using our CCGPMA, we use the framework proposed in [27], which introduces a binary variable $\lambda_n^{(r)} \in \{0,1\}$ representing the r-th labeler's reliability as a function of each sample $\boldsymbol{x}_n$. If $\lambda_n^{(r)} = 1$, the r-th annotator is supposed to provide the actual label, yielding to a categorical distribution. Conversely, $\lambda_n^{(r)} = 0$ indicates that the r-th annotator gives an incorrect output, modeled by a uniform distribution. Therefore, the likelihood function is given as:

$$p(\boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{n=1}^{N} \prod_{r \in R_n} \left(\prod_{k=1}^{K} \zeta_{k,n}^{\delta(y_n^{(r)},k)} \right)^{\lambda_n^{(r)}} \left(\frac{1}{K} \right)^{(1-\lambda_n^{(r)})}, \tag{5.16}$$

where $\delta(y_n^{(r)}, k) = 1$, if $y_n^{(r)} = k$, otherwise $\delta(y_n^{(r)}, k) = 0$. Besides, $\zeta_{k,n} = p(y_n^{(r)} = k|\lambda_n^{(r)} = 1)$ is an estimation of the unknown ground truth. Accordingly, $J = K + R$ LFs are required within our CCGPMA approach, to model the likelihood's parameters $\boldsymbol{\theta}$. In particular, K LFs are used to model $\zeta_{k,n}$ based on a softmax function as:

$$\zeta_{k,n} = \Xi(f_k(\boldsymbol{x}_n)) = \frac{\exp(f_k(\boldsymbol{x}_n))}{\sum_{j=1}^{K} \exp(f_j(\boldsymbol{x}_n))}. \tag{5.17}$$

Besides, R LFs are utilized to compute each $\lambda_n^{(r)}$ from a step function; therefore, $\lambda_n^{(r)} = 1$ if $f_{l_r}(\boldsymbol{x}_n) \geq 0$, otherwise, $\lambda_n^{(r)} = 0$ ($r \in \{1, \dots R\}$). $l_r = K + r \in \{K+1, \dots J\}$ indexes the r-th annotator's LF. It is worth mentioning that we approximate the step function through the well-known sigmoid function ς to avoid discontinuities and favor the CCGPMA implementation. Like to CCGP, we use variational inference to approximate the posterior distribution of our CCGPMA. In consequence, the actual posterior $p(\hat{\boldsymbol{f}}, \boldsymbol{u}|\boldsymbol{Y})$ is approximated following equation (5.13). Besides, we can derive a CCGPMA ELBO, yielding

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{r \in R_n} \mathbb{E}_{q(\boldsymbol{f}_1),\dots,q(\boldsymbol{f}_J)} \left[\log \left(\prod_{k=1}^{K} \zeta_{k,n}^{\delta(y_n^{(r)},k)} \right)^{\lambda_n^{(r)}} \left(\frac{1}{K} \right)^{(1-\lambda_n^{(r)})} \right] - \cdots$$
$$\cdots - \sum_{q=1}^{Q} \mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q)). \tag{5.18}$$

Finally, given a new sample $\boldsymbol{x}_*$, we are interested in the mean and variance for predictive distributions related to the ground truth $\zeta_{k,*} = p(y_* = k)$, and the labelers' reliabilities $\lambda_*^{(r)}$. Accordingly, for $\zeta_{k,*}$ we obtain

$$\mathbb{E}[\zeta_{k*}|\boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] \approx \int \Xi(f_k(\boldsymbol{x}_*))q(\boldsymbol{f}_*)d\boldsymbol{f}_*, \tag{5.19}$$

where $q(\boldsymbol{f}_*) = \int p(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{u})d\boldsymbol{u}$. Similarly, for the predictive variance of $\zeta_{k,*}$, we use the expression $\mathrm{Var}[\zeta_{k,*}] = \mathbb{E}[\zeta_{k,*}^2] - \mathbb{E}[\zeta_{k,*}]^2$; hence, we need to compute $\mathbb{E}[\zeta_{k,*}^2]$ as

$$\mathbb{E}[\zeta_{k*}^2|\boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] \approx \int \Xi(f_k(\boldsymbol{x}_*))^2 q(\boldsymbol{f}_*)d\boldsymbol{f}_*. \tag{5.20}$$

On the other hand, regarding the predictive mean and variance for $\lambda_*^{(r)}$, we have

$$\mathbb{E}[\lambda_*^{(r)}|\boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = \int \varsigma(f_{l_r,*})q(\boldsymbol{f}_*)d\boldsymbol{f}_*. \tag{5.21}$$

For the variance of $\lambda_*^{(r)}$, we use the expression $\mathrm{Var}[\lambda_*^{(r)}] = \mathbb{E}[(\lambda_*^{(r)})^2] - \mathbb{E}[\lambda_*^{(r)}]^2$; hence, we need to compute

$$\mathbb{E}[(\lambda_*^{(r)})^2|\boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = \int \varsigma(f_{l_r,*})^2 q(\boldsymbol{f}_*)d\boldsymbol{f}_*. \tag{5.22}$$

In this case, integrals in equations (5.19) to (5.22) have not closed solution; hence, we approximate them using the Gaussian-Hermite quadrature.

5.2.2 Regression

On the other hand, For real-valued outputs, *e.g.*, $\mathcal{Y}\in\mathbb{R}$, we follow the multi-annotator model used in [1, 22, 23, 52], where each output $y_n^{(r)}$ is considered to be a corrupted version of the hidden ground truth y_n. Then:

$$p(\boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{r\in R_n} \mathcal{N}\left(y_n^{(r)}|y_n, v_n^{(r)}\right), \tag{5.23}$$

where $v_n^{(r)} \in \mathbb{R}^+$ is the r-th annotator error-variance for the instance n. In turn, to model this likelihood function with CCGPMA, it is necessary to chain each likelihood's paramater to a latent function f_j. Thus, we require $J = R + 1$ LFs; one to model the hidden ground truth, such that $y_n = f_1(\boldsymbol{x}_n)$, and R LFs to model each error-variance $v_n^{(r)} = \exp(f_{l_r}(\boldsymbol{x}_n))$, with $r \in \{1, \dots R\}$, and $l_r = r + 1 \in \{2, \dots J\}$. Note that we use an exponential function to map from f_{l_r} to $v_n^{(r)}$, aiming to guarantee $v_n^{(r)} > 0$ ($f_{l_r} \in \mathbb{R}$).

Similar to the classification problem, the actual posterior $p(\hat{\boldsymbol{f}}, \boldsymbol{u}|\boldsymbol{Y})$ is approximated following equation (5.13). Further, the CCGPMA ELBO in regression settings is given by

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{r \in R_n} \mathbb{E}_{q(\boldsymbol{f}_1),\ldots,q(\boldsymbol{f}_J)} \left[\log \mathcal{N}\left(y_n^{(r)} | y_n, v_n^{(r)} \right) \right] - \sum_{q=1}^{Q} \mathbb{D}_{KL}(q(\boldsymbol{u}_q) || p(\boldsymbol{u}_q)). \tag{5.24}$$

Now, given a new sample $\boldsymbol{x}_*$, we are interested in the mean and variances for predictive distributions concerning the ground truth y_*, and the labelers' error-variances $v_*^{(r)}$. First, for y_* we have that since $\mathbf{y} = \mathbf{f}_1$, the posterior distribution for y_* corresponds to $q(f_{1*})$, yielding:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = \mu_{1,*} \tag{5.25}$$

$$\text{Var}[y_* | \boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = s_{1,*}, \tag{5.26}$$

where $\mu_{1,*}$, and $s_{1,*}$ are respectively the mean and variance of $q(f_{1*})$. Then, for $v_*^{(r)}$, we note that due to $\mathbf{v}_r = \exp(\mathbf{f}_{l_r})$, the posterior distribution for $v_*^{(r)}$ follows a log-normal distribution with parameters $\mu_{l_r,*}$ and $s_{l_r,*}$, which correspond to the mean and variance of $q(f_{l_r,*})$ respectively. In this sense, the mean and variance of $v_*^{(r)}$ are given as:

$$\mathbb{E}[v_*^{(r)} | \boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = \exp\left(\mu_{l_r,*} + \frac{s_{l_r,*}}{2} \right). \tag{5.27}$$

$$\text{Var}[v_*^{(r)} | \boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u}] = \exp\left(2\mu_{l_r,*} + s_{l_r,*} \right) \left(\exp(s_{l_r,*}) - 1 \right). \tag{5.28}$$

5.3 Experimental Set-Up

In this section we introduce the experiments' configurations to validate our CCGPMA in both classification and regression settings.

5.3.1 Classification

Testing Datasets

We test our approach using three types of datasets: *fully synthetic data*, *semi-synthetic data*, and *fully real datasets*.

First, we generate *fully synthetic data* as one-dimensional ($P = 1$) multi-class classification problem ($K = 3$). The input feature matrix $\boldsymbol{X}$ is built by randomly sampling $N = 100$ points from an uniform distribution within the interval $[0, 1]$. The true label for the n-th sample is generated by taking the $\arg\max_i \{t_{n,i} : i \in \{1, 2, 3\}\}$, where $t_{n,1} = \sin(2\pi x_n)$, $t_{n,2} = -\sin(2\pi x_n)$, and $t_{n,3} = -\sin(2\pi(x_n + 0.25)) + 0.5$. Besides, the test instances are obtained by extracting 200 equally spaced samples from the interval $[0, 1]$. Second, to control the label generation, we build *semi-synthetic data* from seven datasets of the UCI repository[1] focused on binary and multi class-classification, which are explained in Section 2.3.1 and the Western dataset introduced in Section 4.3.1. Besides, we use additional datasets from UCI repository: Occupancy Detection

[1] http://archive.ics.uci.edu/ml

Data Set–(Occupancy) ($N = 20560$, $P = 7$, $K = 2$), and Skin Segmentation Data Set–(Skin) ($N = 245057$, $P = 4$, $K = 2$). Finally, we evaluate our proposal on two *fully real datasets*, where both the input features and the annotations are captured from real-world problems. Namely, the Voice and Music dataset presented in Section 2.3.1.

Simulated and Provided Labels

Note that the *fully synthetic* and the *semi-synthetic* datasets do not hold real annotations. Therefore, it is necessary to simulate those labels as corrupted versions of the hidden ground truth. Here, the simulations are performed by assuming: i) dependencies among annotators, and ii) the labelers' performance is modeled as a function of the input features. In turn, the generative model of our approach based on SLFM (termed SFLM-C) is used to build the labels, as follows:

- Define Q deterministic functions $\hat{\mu}_q : \mathcal{X} \to \mathbb{R}$, and their combination parameters $\hat{w}_{l_r,q} \in \mathbb{R}$, $\forall r \in R, n \in N$.
- Compute $\hat{f}_{l_r,n} = \sum_{q=1}^{Q} \hat{w}_{l_r,q} \hat{\mu}_q(\hat{x}_n)$, where $\hat{x}_n \in \mathbb{R}$ is the n-th component of $\hat{\boldsymbol{x}} \in \mathbb{R}^N$, being $\hat{\boldsymbol{x}}$ the $1-$D representation of the input features in $\boldsymbol{X}$ by using the well-known t-distributed Stochastic Neighbor Embedding approach [81].
- Calculate $\hat{\lambda}_n^{(r)} = \varsigma(\hat{f}_{l_r,n})$, where $\varsigma(\cdot) \in [0, 1]$ is the sigmoid function.
- Finally, find the r-th label as $y_n^{(r)} = \begin{cases} y_n, & \text{if } \lambda_n^{(r)} \geq 0.5 \\ \tilde{y}_n, & \text{if } \lambda_n^{(r)} < 0.5 \end{cases}$, where $\tilde{y}_n$ is a flipped version of the actual label y_n.

Moreover, aiming to validate our approach under different labels' distributions, we employ the simulation method *Biased coin (Non-homogeneous accuracy)*, which assumes that the annotators' performance depends on the input space (Section 3.3.1).

CCGPMA Training and Method Comparison

The classification performance is assessed as the Area Under the Curve–(AUC) and the overall accuracy (Acc). Further, the AUC is extended for multi-class settings, as authors in [72] discussed. We use a cross-validation scheme with 15 repetitions where 70% of the samples are utilized for training and the remaining 30% for testing (except for the music dataset training and testing sets are clearly defined). Table 5.1 displays the employed methods of the state-of-the-art for comparison purposes. The abreviations are fixed as: Gaussian Processes classifier (GPC), logistic regression classifier (LRC), majority voting (MV), multiple annotators (MA), Modelling annotators expertise (MAE), Learning from crowds (LFC), Distinguishing good from random labelers (DGRL), kernel alignment-based annotator relevance analysis (KAAR).

On the other hand, the Radial basis function–(RBF) kernel is preferred in pattern classification because of its universal approximating ability and mathematical tractability. Hence, for all GP-based approaches, the kernel functions are fixed as:

Table 5.1 A brief overview of the state-of-the-art methods tested

Algorithm	Description
GPC-GOLD	A GPC using the real labels (upper bound).
GPC-MV	A GPC using the MV of the labels as the ground truth.
MA-LFC-C [1]	A LRC with constant parameters across the input space.
MA-DGRL [27]	A multi-labeler approach that considers as latent variables the annotator performance.
MA-GPC [20]	A multi-labeler GPC, which is as an extension of MA-LFC.
MA-GPCV [49]	An extension of MA-GPC that includes variational inference and priors over the labelers' parameters.
MA-DL [25]	A Crowd Layer for DL, where the annotators' parameters are constant across the input space.
KAAR [54]	A kernel-based approach that employs a convex combination of classifiers and codes labelers dependencies.
CGPMA-C	A particular case of our CCGPMA for classification, where $Q = J$, and we fix $w_{j,q} = 1$, if $j = q$, otherwise $w_{j,q} = 0$.

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) = s^2 \exp\left(\frac{-||\boldsymbol{x}_n - \boldsymbol{x}_{n'}||_2^2}{2l^2}\right), \tag{5.29}$$

where $||\cdot||^2$ stands for the L2 norm, $n, n' \in \{1, 2, \ldots, N\}$, and $s, l \in \mathbb{R}^+$ are the kernel hyper-parameters. For concrete testing, we fix $s = 1$, while l is estimated by optimizing the corresponding ELBO in equation (5.18) (we use a gradient-based optimization. The required gradients, and the predictive distributions are presented in Section A.2.1). Moreover, for CGPMA, we fix $Q = R + K$, since each LF $f_j(\cdot)$ is linked to $u_q(\cdot)$. On the other hand, for CCGPMA, each $f_j(\cdot)$ is built as a convex combination of $\mu_q(\cdot)$ (equation (5.9)); therefore, there is no restriction concerning Q. However, to make a fair comparison with CGPMA, we also fix $Q = R + K$ in CCGPMA. For the *fully synthetic datasets*, we use $M = 10$ inducing points per latent function, and for the remaining experiments, we test with $M = 40$, and $M = 80$. For all the experiments, we use stochastic inference with a mini-batch size of 100. The CCGPMA's Python code is publicly available.[2]

5.3.2 Regression

Testing Datasets

Aiming to test our CCGPMA in regression scenarios, we use the synthetic, semi-synthetic, and real dataset presented in Section 2.3.2.

Simulated and Provided Labels

For *fully synthetic* and *semi-synthetic* datasets do not hold real annotations. Thus, it is necessary to generate these labels synthetically as a version of the gold standard corrupted by Gaussian

[2]https://github.com/juliangilg/CCGPMA

noise, *i.e.*, $y_n^{(r)} = y_n + \epsilon_n^{(r)}$, where $\epsilon_n^{(r)} \sim \mathcal{N}(0, v_n^{(r)})$, being $v_n^{(r)}$ the r-th annotator error-variance for the sample n. Note that we are interested in modeling such an error-variance for the r-th annotator as a function of the input features, correlated with the other labelers' variances. In turn, an SLFM-based approach is used to build the labels, as follows:

- Define Q functions $\hat{\mu}_q : \mathcal{X} \to \mathbb{R}$, and the combination parameters $\hat{w}_{l_r,q} \in \mathbb{R}$, $\forall r, q$.
- Compute $\hat{f}_{l_r,n} = \sum_{q=1}^{Q} \hat{w}_{l_r,q} \hat{\mu}_q(\hat{x}_n)$, where $\hat{x}_n$ is the n-th component of $\hat{\boldsymbol{x}} \in \mathbb{R}$, which is an $1-$D representation of input features $\boldsymbol{X}$ by using the t-distributed Stochastic Neighbor Embedding approach [81].
- Finally, determine $\hat{v}_n^{(r)} = \exp(\hat{f}_{l_r,n})$.

Besides, aiming to validate our approach using different labels' distributions, we use the simulation method described in Section 3.3.2 (termed *Non-homogeneous error-variance*), which assumes that the annotators' performance depends on the input space.

CCGPMA Training and Method Comparison

The quality assessment is carried out by estimating the regression performance as the coefficient of determination–(R^2) and Pearson correlation coefficient–(Pear). A cross-validation scheme is employed with 15 repetitions where 70% of the samples are utilized for training and the remaining 30% for testing (except for *fully synthetic dataset*, since it clearly defines the training and testing sets). Table 2.4 displays the employed methods of the state-of-the-art for comparison purposes.

Besides, for CGPMA, we set $Q = R+1$, where each LF $f_j(\cdot)$ is linked to $u_q(\cdot)$. On the other hand, for CCGPMA, each $f_j(\cdot)$ is built as a convex combination of $\mu_q(\cdot)$ (equation (5.9)); therefore, there is no restriction concerning Q. However, to make a fair comparison with CGPMA, we also fix $Q = R + 1$ in CCGPMA. For the *fully synthetic datasets*, we use $M = 10$ inducing points per latent function, and for the remaining experiments, we test with $M = 40$, and $M = 80$. For all the experiments, we use stochastic inference with a mini-batch size of 100. The model parameters are estimated by optimizing the ELBO in equation (5.24). Such optimization is performed via a gradient-based algorithm (The required gradients, and the predictive distributions are presented in Chapter A)

5.4 Results and Discussion

5.4.1 Classification

We first perform a controlled experiment to test the CCGPMA capability when dealing with binary and multi-class classification. We use the *fully synthetic* dataset described in Section 5.3.1. Besides, five labelers ($R = 5$) are simulated with different levels of expertise. To simulate the annotators' performance based on the method SLFM-C, we define $Q = 3$ $\hat{\mu}_q(\cdot)$ functions,

yielding:

$$\hat{\mu}_1(x) = 4.5\cos(2\pi x + 1.5\pi) - 3\sin(4.3\pi x + 0.3\pi), \tag{5.30}$$

$$\hat{\mu}_2(x) = 4.5\cos(1.5\pi x + 0.5\pi) + 5\sin(3\pi x + 1.5\pi), \tag{5.31}$$

$$\hat{\mu}_3(x) = 1, \tag{5.32}$$

where $x \in [0, 1]$. Besides, the combination weights are gathered within the following combination matrix $\hat{\boldsymbol{W}} \in \mathbb{R}^{Q\times R}$:

$$\hat{\boldsymbol{W}} = \begin{bmatrix} 0.4 & 0.7 & -0.5 & 0.0 & -0.7 \\ 0.4 & -1.0 & -0.1 & -0.8 & 1.0 \\ 3.1 & -1.8 & -0.6 & -1.2 & 1.0 \end{bmatrix}, \tag{5.33}$$

holding elements $\hat{w}_{l_r,q}$. Similarly, For the *Non-homogeneous accuracy* approach, we divide the input space into five regions and define the performance matrix $\boldsymbol{P} \in [0, 1]^{R\times R}$ defined by equation (4.10).

For visual inspection purposes, we perform an initial experiment using the simulation method SLFM-C. Figure 5.1 shows the predictive label's probability–(PLP), $p(y_* = k|\boldsymbol{x}_*)$, and the AUC for all studied approaches regarding the *fully synthetic* data. Notice that for methods MA-GPC, MA-GPCV, and KAAR, we use the *one-vs-all* scheme for this experiment (such methods were defined only for binary classification settings). Accordingly, the PLP corresponds to scores rather than probabilities for those models. Besides, regarding the PLP of our CGPMA and CCGPMA, we provide the mean and variance for the predictive distribution $\zeta_{k,*} = p(y_* = k|\boldsymbol{x}_*, \hat{\boldsymbol{f}}, \boldsymbol{u})$, which are computed based on equations (5.19) and (5.20). As seen in Figure 5.1, KAAR, MA-GPC, and MA-GPCV present a different shape than the ground truth; moreover, KAAR and MA-GPCV exhibit the worst AUC, even worse than the intuitive lower bound GPC-MV. We explain such conduct because these approaches are designed to deal with binary labels [19, 20, 54]. To face such a problem, we use the *one-vs-all* scheme; still, it can lead to ambiguously classified regions [9]. We note an akin predictive AUC concerning MA-DL methods and the linear approaches MA-LFC-C and MA-DGRL. Nonetheless, the linear techniques exhibit a PLP less similar to the Ground truth, because MA-LFC-C and MA-DGRL only deal with linearly separable data. Further, we analyze the results of our CGPMA-C and its particular enhancement CCGPMA-C. Our methods' predictive AUC is close to deep learning and linear models. Unlike them, our CGPMA-C and CCGPMA-C show the most accurate PLP compared to the absolute gold standard. CCGPMA-C behaves quite similarly to GPC-GOLD, which is the theoretical upper bound. Finally, from the GPC-MV, we do not identify notable differences with the rest of the approaches (excluding KAAR and MA-GPCV).

From the above, we recognize that by analyzing both the predictive AUC and the PLP, our

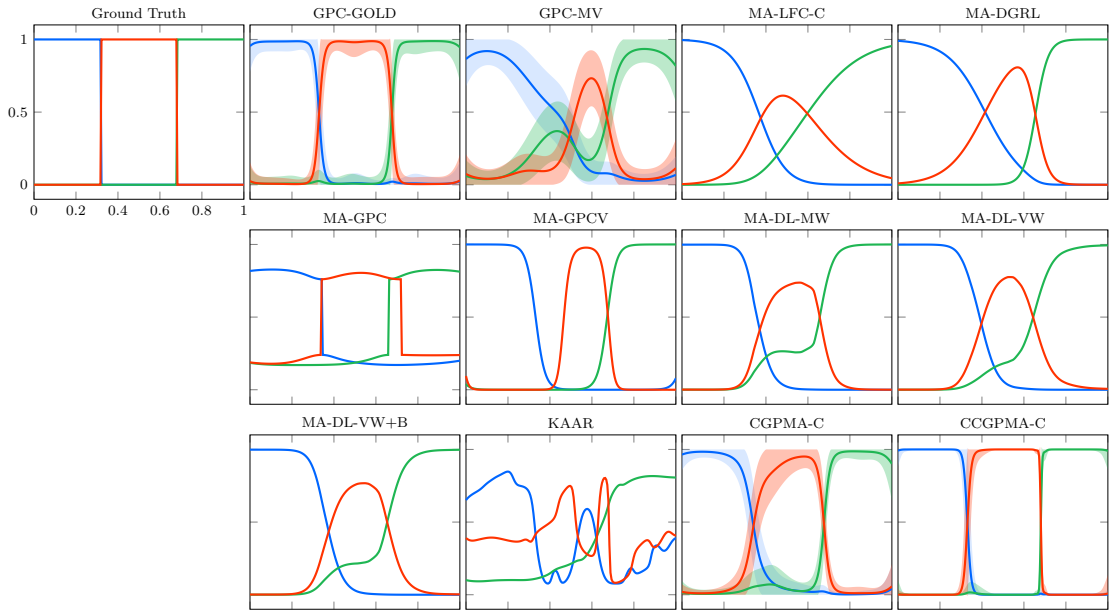

Figure 5.1 Fully synthetic dataset results. The PLP is shown, comparing the prediction of our CCGPMA-C ($AUC = 1$) and CCGPMA-C ($AUC = 0.9999$) against: the theoretical upper bound GPC-GOLD ($AUC = 1.0$), the lower bound GPC-MV($AUC = 0.9809$), and the state-of-the-art approaches MA-LFC-C ($AUC = 0.9993$), MA-DGRL ($AUC = 0.9999$), MA-GPC ($AUC = 0.9977$), MA-GPCV ($AUC = 0.9515$), MA-DL-MW ($AUC = 0.9989$), MA-DL-VW ($AUC = 0.9972$), MA-DL-VW+B ($AUC = 0.9994$), KAAR (0.9099). The shaded region in GPC-MV, CGPMA-C, and CCGPMA-C indicates the area enclosed by the mean $\pm$ two standard deviations. There is no shaded region for approaches lacking prediction uncertainty

CCGPMA-C exhibits the best performance obtaining similar results compared to the intuitive upper bound (GPC-GOLD). Accordingly, CCGPMA-C proffers a more suitable representation of the labelers' behavior than its competitors. Indeed, CCGPMA-C codes both the annotators' dependencies and the relationship between the input features and the annotators' performance. To empirically support the above statement, Figure 5.2 shows the estimated per-annotator reliability, where we consider models that include such parameters (MA-DGRL, CGPMA-C, and CCGPMA-C). As seen, MA-DGRL (column 2 in Figure 5.2) does not offer a proper representation of the annotators' behavior. CGPMA-C and CCGPMA-C (columns 3 and 4 in Figure 5.2) outperform MA-DGRL, which is a direct repercussion of modeling the labelers' parameters as functions of the input features. We observe that CCGPMA-C exhibits the best performance in terms of accuracy; such an outcome is due to this method improving the quality of the annotators' model by considering correlations among their decisions [10, 54]).

It is worth mentioning that semi-synthetic experiments are a common practice in the *learning from crowds* area [19, 49], where the input features come from real-world datasets whilst the labels from multiple annotators are simulated following the *fully synthetic data* set-up (see equations (5.30) to (5.33)). Table 5.2 (a) shows the results concerning semi-synthetic datasets using the simulation method SLFM-C. On average, our CCGPMA-C accomplishes the best predictive AUC; moreover, we note that CGPMA-C reaches the second-best performance. Furthermore,

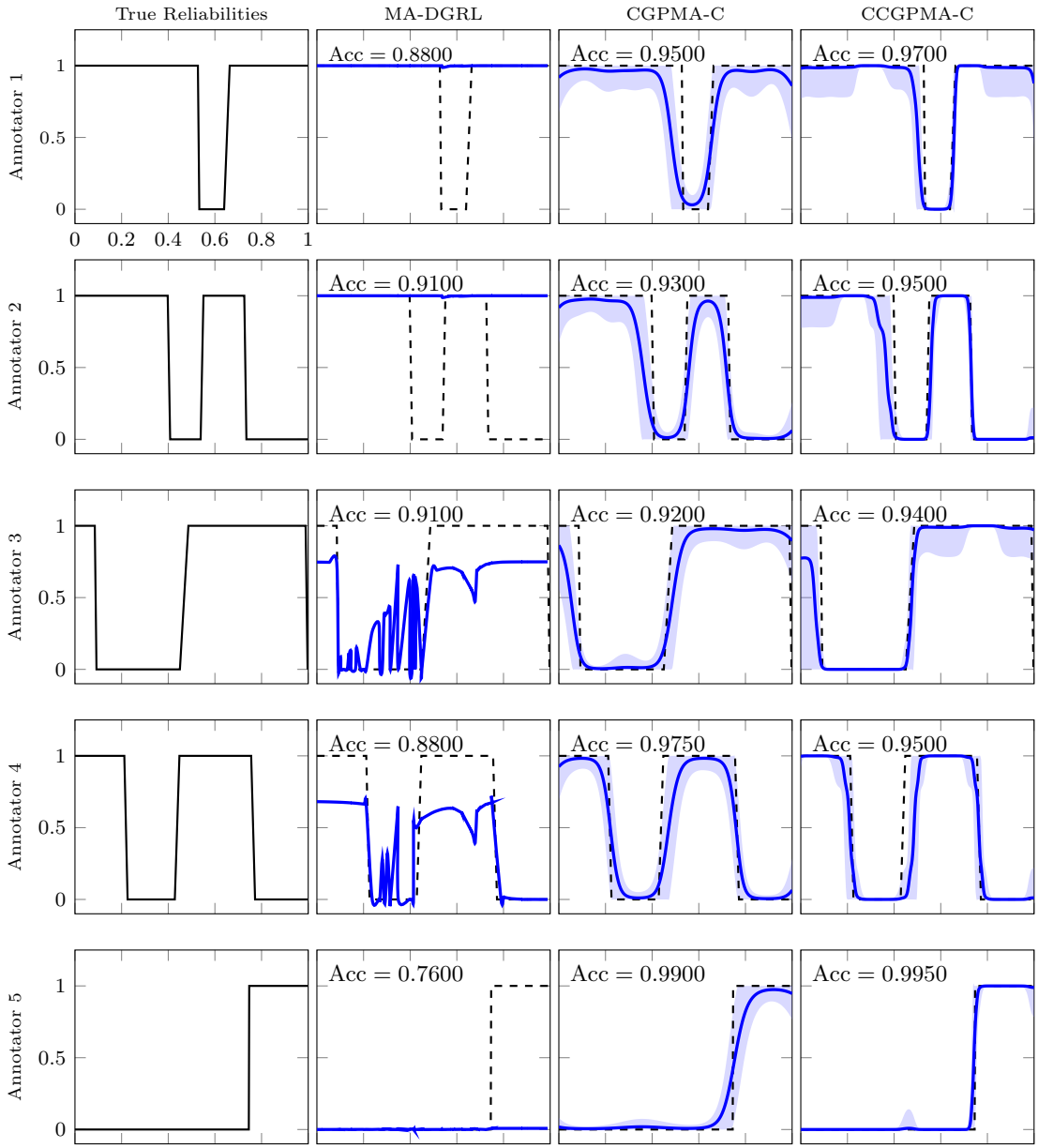

Figure 5.2 Fully synthetic data reliability results. From top to bottom, the first column exposes the true reliabilities (λ_r). The subsequent columns present the estimation of the reliabilities performed by state-of-the-art models, where the correct values are provided in dashed lines. The shaded region in CGPMA-C and CCGPMA-C indicates the area enclosed by the mean $\pm$ two standard deviations. Also, the accuracy (Acc) is provided

the GPs-based competitors achieve competitive results (GPC-MV, MA-GPC, MA-GPCV, and KAAR). On the other hand, the GPC-MV method obtains a significantly lower performance than our CCGPMA-C, which is explained because GPC-MV is the most naive approach since it considers that the whole annotators exhibit the same performance. Conversely, analyzing the results from MA-GPC, MA-GPCV, and KAAR, we note that they perform worse than GPC-MV. We explain such an outcome in two ways. First, these approaches do not model the relationship between the input features and the annotators' performance. Second, as exposed in a previous experiment

Table 5.2 Semi-synthetic classification results

(a) SLFM-C

Method		Breast	Bupa	Ionosphere	Pima	TicTacToe	Occupancy	Skin	Western	Wine	Segmentation	Average
GPC-GOLD($M=40$)	AUC[%]	99.07 ± 0.45	69.75 ± 4.66	94.90 ± 2.35	83.78 ± 3.02	84.29 ± 3.34	99.56 ± 0.06	99.97 ± 0.01	91.85 ± 0.61	99.87 ± 0.15	95.96 ± 1.96	91.90
	Acc[%]	96.62 ± 1.00	65.38 ± 4.03	85.91 ± 3.61	77.92 ± 2.48	79.61 ± 2.55	98.83 ± 0.09	99.62 ± 0.13	72.09 ± 1.01	96.67 ± 2.24	75.26 ± 8.69	84.79
GPC-GOLD($M=80$)	AUC[%]	99.03 ± 0.46	69.97 ± 4.83	95.13 ± 2.25	83.74 ± 2.97	84.91 ± 3.23	99.56 ± 0.06	99.97 ± 0.01	92.50 ± 0.57	99.88 ± 0.16	97.81 ± 0.41	92.25
	Acc[%]	96.62 ± 1.07	66.09 ± 3.94	86.16 ± 3.36	77.84 ± 2.51	80.07 ± 2.47	98.83 ± 0.10	99.58 ± 0.11	73.00 ± 1.30	96.79 ± 2.26	86.57 ± 1.87	86.15
GPC-MV($M=40$)	AUC[%]	98.97 ± 0.45	53.66 ± 5.16	75.66 ± 5.72	53.99 ± 7.60	66.20 ± 3.57	75.85 ± 19.16	84.58 ± 0.90	86.58 ± 3.31	81.79 ± 2.12	95.62 ± 2.28	77.29
	Acc[%]	94.93 ± 1.27	49.42 ± 7.21	64.91 ± 3.93	66.81 ± 2.80	62.36 ± 2.42	80.40 ± 1.84	82.64 ± 0.71	60.81 ± 5.05	59.51 ± 6.22	75.62 ± 9.54	69.74
GPC-MV($M=80$)	AUC[%]	98.92 ± 0.48	56.98 ± 5.29	77.79 ± 5.50	53.02 ± 6.74	67.44 ± 3.57	63.12 ± 19.68	84.20 ± 0.80	84.46 ± 0.89	83.23 ± 4.87	97.49 ± 0.47	76.66
	Acc[%]	94.96 ± 1.45	54.42 ± 5.51	64.91 ± 3.93	67.73 ± 1.46	63.12 ± 2.17	78.41 ± 2.38	83.06 ± 0.33	57.64 ± 1.80	61.73 ± 11.56	84.00 ± 1.67	71.00
MA-LFC-C	AUC[%]	87.89 ± 5.10	45.93 ± 14.44	73.58 ± 9.01	81.19 ± 3.13	60.04 ± 2.61	89.42 ± 0.79	$\mathbf{94.40 \pm 0.08}$	84.00 ± 2.11	96.92 ± 3.57	98.92 ± 0.31	81.23
	Acc[%]	67.71 ± 2.85	46.54 ± 11.32	62.96 ± 6.01	76.62 ± 2.21	60.39 ± 2.48	70.83 ± 0.78	$\mathbf{93.52 \pm 0.29}$	59.71 ± 1.41	90.25 ± 5.12	92.17 ± 0.78	72.07
MA-DGRL	AUC[%]	97.57 ± 1.89	57.24 ± 3.36	64.53 ± 7.21	81.38 ± 2.90	61.29 ± 2.30	49.71 ± 1.05	93.79 ± 1.07	81.43 ± 1.50	97.95 ± 2.21	98.97 ± 0.38	78.39
	Acc[%]	89.92 ± 3.28	53.72 ± 3.84	33.27 ± 3.01	77.00 ± 2.71	62.06 ± 2.52	66.73 ± 0.46	89.30 ± 1.01	51.21 ± 1.58	89.01 ± 4.28	93.27 ± 1.05	70.55
MA-GPC	AUC[%]	98.11 ± 1.16	54.46 ± 5.78	66.31 ± 14.74	53.25 ± 17.80	60.79 ± 9.95	92.57 ± 7.96	80.89 ± 0.60	86.71 ± 1.14	94.17 ± 2.62	97.34 ± 0.35	78.46
	Acc[%]	96.13 ± 1.07	50.38 ± 4.69	62.14 ± 11.13	52.41 ± 14.30	58.70 ± 8.58	60.62 ± 1.99	76.70 ± 0.46	72.18 ± 1.61	90.49 ± 3.42	88.86 ± 2.10	70.86
MA-GPCV	AUC[%]	82.70 ± 5.47	55.67 ± 6.83	62.38 ± 8.71	62.17 ± 5.90	61.04 ± 10.03	60.22 ± 2.66	76.29 ± 3.74	84.51 ± 1.47	97.35 ± 1.72	99.24 ± 0.27	74.16
	Acc[%]	68.72 ± 3.24	52.37 ± 6.67	60.94 ± 5.64	61.21 ± 5.83	65.69 ± 2.33	56.25 ± 3.01	76.67 ± 2.30	69.45 ± 1.48	94.20 ± 3.35	95.95 ± 1.19	70.15
MA-DL-MW	AUC[%]	94.70 ± 1.73	52.37 ± 5.68	75.35 ± 5.43	61.78 ± 2.67	68.27 ± 2.96	64.09 ± 2.26	86.36 ± 0.57	90.92 ± 0.56	97.28 ± 1.09	99.50 ± 0.17	79.06
	Acc[%]	90.24 ± 4.52	51.67 ± 6.17	$\mathbf{70.25 \pm 3.90}*$	55.12 ± 3.64	62.87 ± 2.83	69.01 ± 1.30	46.03 ± 0.26	62.70 ± 1.05	88.52 ± 3.30	93.91 ± 1.33	69.03
MA-DL-VW	AUC[%]	95.26 ± 2.45	53.27 ± 6.18	69.87 ± 4.97	60.63 ± 3.36	67.71 ± 2.67	68.40 ± 3.45	86.56 ± 0.68	91.73 ± 0.67	98.07 ± 1.52	$\mathbf{99.72 \pm 0.11}$	79.12
	Acc[%]	90.57 ± 1.66	51.67 ± 5.65	67.30 ± 4.90	54.46 ± 1.97	64.42 ± 2.47	63.99 ± 1.71	46.19 ± 0.29	68.98 ± 1.48	91.23 ± 4.77	$\mathbf{95.89 \pm 0.92}*$	69.47
MA-DL-VW+B	AUC[%]	94.65 ± 2.42	52.81 ± 6.31	71.96 ± 4.53	61.23 ± 3.78	67.80 ± 3.42	67.82 ± 3.86	86.68 ± 0.67	91.64 ± 0.85	98.17 ± 1.55	$\mathbf{99.72 \pm 0.09}$	79.25
	Acc[%]	90.37 ± 1.88	52.31 ± 5.53	67.92 ± 4.76	56.68 ± 3.70	62.52 ± 3.28	64.15 ± 2.36	46.04 ± 0.28	64.16 ± 1.85	90.74 ± 4.75	95.65 ± 1.10	69.05
KAAR	AUC[%]	80.58 ± 2.74	59.20 ± 6.63	70.46 ± 7.39	58.02 ± 4.06	63.81 ± 5.45	69.16 ± 2.06	51.58 ± 4.74	85.88 ± 1.20	99.43 ± 1.05	92.17 ± 1.90	73.03
	Acc[%]	70.18 ± 3.58	$\mathbf{56.22 \pm 5.20}*$	63.14 ± 6.68	61.44 ± 2.64	56.32 ± 2.61	53.62 ± 1.10	39.84 ± 2.78	62.35 ± 1.86	93.58 ± 4.53	77.85 ± 5.82	63.45
CGPMA-C($M=40$)	AUC[%]	99.20 ± 0.38	57.13 ± 4.68	83.56 ± 10.02	82.01 ± 3.14	70.56 ± 3.04	82.20 ± 2.73	92.62 ± 1.20	91.78 ± 0.66	99.82 ± 0.18	96.79 ± 0.65	85.56
	Acc[%]	96.10 ± 1.26	47.95 ± 7.36	65.66 ± 3.70	76.83 ± 2.81	67.62 ± 2.51	61.15 ± 1.56	90.62 ± 1.34	71.13 ± 1.83	95.31 ± 2.41	82.22 ± 2.20	75.46
CGPMA-C($M=80$)	AUC[%]	99.14 ± 0.38	56.96 ± 4.74	86.15 ± 6.96	$\mathbf{82.04 \pm 3.18}$	70.48 ± 3.12	$\mathbf{99.08 \pm 0.26}$	90.46 ± 1.64	91.85 ± 0.57	$\mathbf{99.84 \pm 0.12}$	94.06 ± 0.61	87.01
	Acc[%]	95.06 ± 1.07	51.47 ± 5.99	65.16 ± 3.95	$\mathbf{76.97 \pm 2.31}*$	66.97 ± 2.93	85.75 ± 0.88	85.32 ± 1.50	70.66 ± 1.27	$\mathbf{95.56 \pm 1.95}*$	71.43 ± 3.47	76.43
CCGPMA-C($M=40$)	AUC[%]	$\mathbf{99.38 \pm 0.27}$	$\mathbf{60.22 \pm 5.06}$	87.84 ± 6.72	78.10 ± 6.22	$\mathbf{74.95 \pm 5.39}$	91.98 ± 2.00	85.70 ± 2.66	93.09 ± 0.51	99.44 ± 0.33	97.67 ± 0.53	86.84
	Acc[%]	96.68 ± 1.23	50.71 ± 6.73	65.72 ± 4.43	67.13 ± 11.17	$\mathbf{67.64 \pm 5.45}*$	87.08 ± 1.79	82.39 ± 0.96	73.63 ± 1.10	93.46 ± 3.04	84.61 ± 3.21	76.90
CCGPMA-C($M=80$)	AUC[%]	99.33 ± 0.30	59.19 ± 5.65	$\mathbf{90.55 \pm 6.29}$	80.45 ± 5.10	73.12 ± 3.23	97.75 ± 2.00	89.42 ± 2.20	$\mathbf{93.15 \pm 0.50}$	99.43 ± 0.33	97.58 ± 0.43	$\mathbf{88.00}$
	Acc[%]	$\mathbf{97.17 \pm 0.87}*$	51.86 ± 5.54	65.97 ± 4.18	67.56 ± 14.16	65.30 ± 3.11	$\mathbf{91.56 \pm 5.64}*$	86.84 ± 1.99	$\mathbf{73.98 \pm 1.55}*$	92.96 ± 3.30	83.56 ± 4.17	$\mathbf{77.68}*$

(b) Non-homogeneous accuracy

Method		Breast	Bupa	Ionosphere	Pima	TicTacToe	Occupancy	Skin	Western	Wine	Segmentation	Average
GPC-GOLD($M=40$)	AUC[%]	99.07 ± 0.45	69.75 ± 4.66	94.90 ± 2.35	83.78 ± 3.02	84.29 ± 3.34	99.56 ± 0.06	99.97 ± 0.01	91.85 ± 0.61	99.87 ± 0.15	95.96 ± 1.96	91.90
	Acc[%]	96.62 ± 1.00	65.38 ± 4.03	85.91 ± 3.61	77.92 ± 2.48	79.61 ± 2.55	98.83 ± 0.09	99.62 ± 0.13	72.09 ± 1.01	96.67 ± 2.24	75.26 ± 8.69	84.79
GPC-GOLD($M=80$)	AUC[%]	99.03 ± 0.46	69.97 ± 4.83	95.13 ± 2.25	83.74 ± 2.97	84.91 ± 3.23	99.56 ± 0.06	99.97 ± 0.01	92.50 ± 0.57	99.88 ± 0.16	97.81 ± 0.41	92.25
	Acc[%]	96.62 ± 1.07	66.09 ± 3.94	86.16 ± 3.36	77.84 ± 2.51	80.07 ± 2.47	98.83 ± 0.10	99.58 ± 0.11	73.00 ± 1.30	96.79 ± 2.26	86.57 ± 1.87	86.15
GPC-MV($M=40$)	AUC[%]	93.23 ± 6.06	63.66 ± 8.50	76.47 ± 15.09	66.27 ± 15.45	64.57 ± 2.64	60.90 ± 20.43	54.09 ± 12.70	90.27 ± 0.55	98.53 ± 1.60	95.40 ± 2.11	76.34
	Acc[%]	91.67 ± 4.19	57.24 ± 5.96	70.06 ± 12.75	69.44 ± 7.35	63.01 ± 2.28	79.59 ± 3.19	25.28 ± 16.83	68.86 ± 1.58	82.10 ± 9.94	69.32 ± 9.29	67.66
GPC-MV($M=80$)	AUC[%]	92.89 ± 6.23	63.88 ± 8.19	79.65 ± 11.68	61.61 ± 15.74	66.18 ± 2.75	56.55 ± 17.28	57.72 ± 17.30	91.57 ± 0.48	98.56 ± 1.62	97.18 ± 1.27	76.58
	Acc[%]	91.12 ± 4.45	56.86 ± 5.03	$\mathbf{73.84 \pm 8.35}*$	68.31 ± 11.19	63.50 ± 2.20	78.30 ± 3.52	29.50 ± 22.09	71.11 ± 1.45	83.21 ± 9.51	78.54 ± 9.38	69.43
MA-LFC-C	AUC[%]	77.96 ± 3.82	64.90 ± 6.59	74.70 ± 4.40	$\mathbf{81.90 \pm 2.36}$	54.94 ± 2.92	85.01 ± 4.96	82.45 ± 2.51	85.24 ± 0.97	98.86 ± 1.03	99.05 ± 0.37	80.50
	Acc[%]	72.26 ± 6.65	$\mathbf{62.88 \pm 6.05}*$	75.91 ± 4.28	$\mathbf{76.19 \pm 2.40}*$	56.85 ± 2.92	69.44 ± 4.08	55.25 ± 1.97	55.64 ± 1.44	94.07 ± 2.73	93.58 ± 1.20	71.21
MA-DGRL	AUC[%]	56.88 ± 5.65	$\mathbf{66.41 \pm 7.39}$	57.18 ± 8.85	83.11 ± 1.92	54.56 ± 3.08	70.06 ± 14.56	61.12 ± 1.24	74.22 ± 1.68	93.17 ± 3.55	$\mathbf{99.89 \pm 0.13}$	71.66
	Acc[%]	59.19 ± 4.56	62.12 ± 6.70	65.09 ± 4.97	77.75 ± 2.28	59.56 ± 4.28	41.43 ± 24.82	43.57 ± 0.36	26.09 ± 1.27	63.58 ± 4.57	28.03 ± 1.61	52.64
MA-GPC	AUC[%]	40.52 ± 6.09	45.19 ± 6.26	46.84 ± 5.68	40.88 ± 3.82	57.93 ± 7.38	34.14 ± 3.27	53.13 ± 13.53	86.53 ± 0.85	89.83 ± 3.19	97.69 ± 0.49	53.48
	Acc[%]	58.63 ± 5.45	49.36 ± 4.14	30.06 ± 4.16	56.85 ± 4.22	60.58 ± 5.93	26.67 ± 0.99	57.55 ± 11.06	75.41 ± 0.75	84.07 ± 4.13	92.27 ± 0.92	53.09
MA-GPCV	AUC[%]	62.95 ± 3.77	59.88 ± 5.32	64.12 ± 5.34	65.27 ± 3.44	55.37 ± 5.35	27.93 ± 2.26	55.48 ± 9.45	89.34 ± 0.96	88.91 ± 7.83	98.94 ± 0.35	66.82
	Acc[%]	59.22 ± 2.44	57.37 ± 4.31	61.19 ± 4.51	64.99 ± 2.80	53.06 ± 3.64	25.22 ± 0.43	49.12 ± 11.53	72.12 ± 1.67	83.95 ± 7.30	95.06 ± 1.15	62.13
MA-DL-MW	AUC[%]	81.95 ± 5.95	53.06 ± 4.27	75.77 ± 6.44	68.72 ± 2.81	68.94 ± 2.57	72.50 ± 1.59	44.23 ± 1.79	93.32 ± 0.54	96.07 ± 1.79	99.63 ± 0.24	75.42
	Acc[%]	70.31 ± 5.58	52.05 ± 3.58	71.38 ± 5.96	65.92 ± 3.04	63.68 ± 2.99	75.05 ± 0.48	52.82 ± 0.27	73.34 ± 1.72	85.93 ± 4.01	$\mathbf{95.59 \pm 1.24}*$	70.61
MA-DL-VW	AUC[%]	83.06 ± 6.45	52.84 ± 4.49	73.22 ± 7.61	65.76 ± 3.91	68.67 ± 3.25	65.54 ± 12.99	43.92 ± 1.65	93.45 ± 0.39	95.98 ± 1.96	99.69 ± 0.17	74.21
	Acc[%]	72.13 ± 5.13	51.99 ± 4.20	69.31 ± 7.22	64.27 ± 2.83	62.99 ± 3.74	65.00 ± 20.34	52.68 ± 0.32	$\mathbf{75.53 \pm 0.87}*$	86.67 ± 4.60	95.57 ± 1.22	69.61
MA-DL-VW+B	AUC[%]	83.55 ± 5.23	53.29 ± 4.88	74.51 ± 5.76	66.45 ± 3.53	68.81 ± 3.14	54.14 ± 19.10	43.90 ± 1.61	$\mathbf{93.49 \pm 0.46}$	95.90 ± 1.80	99.66 ± 0.26	73.37
	Acc[%]	69.27 ± 7.52	52.69 ± 3.51	69.31 ± 6.85	64.56 ± 3.36	63.80 ± 4.78	52.32 ± 24.94	52.75 ± 0.38	75.05 ± 1.68	85.56 ± 4.27	95.60 ± 1.69	68.09
KAAR	AUC[%]	66.66 ± 4.06	54.45 ± 9.87	75.43 ± 4.52	73.77 ± 9.17	50.00 ± 0.00	82.84 ± 2.11	92.26 ± 1.44	84.67 ± 1.53	97.85 ± 1.89	96.30 ± 0.68	77.42
	Acc[%]	59.74 ± 3.59	53.65 ± 9.66	73.27 ± 5.61	70.62 ± 4.34	33.52 ± 2.41	72.21 ± 3.05	57.22 ± 10.69	63.55 ± 1.81	90.49 ± 4.19	80.27 ± 3.31	65.45
CGPMA-C($M=40$)	AUC[%]	81.02 ± 9.70	61.31 ± 5.68	81.70 ± 3.87	81.88 ± 1.90	69.01 ± 2.21	70.00 ± 6.95	$\mathbf{96.77 \pm 0.84}$	89.55 ± 0.67	99.61 ± 0.40	63.20 ± 5.94	79.41
	Acc[%]	71.58 ± 8.86	58.78 ± 4.42	67.11 ± 4.94	75.96 ± 2.63	64.98 ± 2.95	29.01 ± 0.67	$\mathbf{87.66 \pm 0.36}*$	65.76 ± 2.17	88.40 ± 9.58	28.52 ± 2.13	63.78
CGPMA-C($M=80$)	AUC[%]	81.42 ± 8.07	61.71 ± 5.09	83.38 ± 3.49	81.36 ± 1.78	$\mathbf{71.66 \pm 2.16}$	67.74 ± 4.01	84.82 ± 1.60	89.48 ± 0.65	99.32 ± 0.53	67.59 ± 1.28	78.85
	Acc[%]	69.72 ± 3.89	59.29 ± 4.23	70.00 ± 5.06	75.67 ± 2.79	$\mathbf{67.38 \pm 2.12}*$	45.52 ± 0.48	62.81 ± 0.41	67.62 ± 1.48	94.44 ± 3.43	15.94 ± 1.63	62.84
CCGPMA-C($M=40$)	AUC[%]	94.63 ± 8.06	61.04 ± 5.00	84.28 ± 3.40	81.34 ± 2.51	60.73 ± 3.51	$\mathbf{96.97 \pm 4.95}$	85.50 ± 13.37	92.56 ± 0.41	$\mathbf{99.75 \pm 0.27}$	98.00 ± 0.38	$\mathbf{85.48}$
	Acc[%]	86.08 ± 11.94	59.29 ± 4.53	70.19 ± 5.13	75.79 ± 2.97	61.48 ± 4.71	$\mathbf{88.84 \pm 13.85}*$	67.77 ± 17.41	73.13 ± 1.40	$\mathbf{96.30 \pm 2.21}$	85.06 ± 3.17	76.39
CCGPMA-C($M=80$)	AUC[%]	$\mathbf{97.00 \pm 6.08}$	61.32 ± 4.43	$\mathbf{86.59 \pm 3.01}$	80.94 ± 2.29	59.50 ± 2.67	85.21 ± 4.68	89.20 ± 13.78	92.85 ± 0.46	99.74 ± 0.27	97.43 ± 2.61	84.98
	Acc[%]	$\mathbf{91.74 \pm 9.74}*$	59.04 ± 3.81	72.26 ± 5.98	75.70 ± 2.87	62.80 ± 2.52	77.92 ± 4.03	72.96 ± 22.32	73.70 ± 1.37	95.80 ± 2.57	83.70 ± 9.69	$\mathbf{76.56}*$

Note: the highest AUC excluding the upper bound (target) classifier GPC-GOLD. Marked with $*$: the highest accuracy (Acc) except the upper bound.

MA-GPC, MA-GPCV, and KAAR use a *one-vs-all* to deal with multi-class problems, which can lead to ambiguously classified regions [9]. The latter can be confirmed in the results for the multi-class dataset "Western" ($K = 4$), where the predictive AUC for such approaches are the lowest. Then, analyzing the results from the DL-based strategies, we note a slightly better performance than the GPs-based methods (excluding CGPMA-C and CCGPMA-C). However, the DL-based performs considerably worse than our proposal because the CrowdLayer provides straightforward codification of the labelers' performance to guarantee a low computational cost [49]. Fi-

nally, from the linear models, we first analyze the outstanding performance of MA-DGRL, which defeats all its non-linear competitors. In particular, the simulated labels (Section 5.3.1) follow the MA-DGRL model, favoring its performance. Though MA-LFC-C achieves competitive performance compared to the DL-based methods, it is considerably lower than our proposal. The MA-LFC-C formulation assumes that the annotators' behavior is homogeneous across the input space, which does not correspond to the labels simulation procedure.

On the other hand, Table 5.2(b) shows the results concerning the simulation method "Non-homogeneous accuracy". At first sight, there exists a generalized lower performance compared to previous results in Table 5.2(a); the above indicates that the annotators' performance is lower, which impacts the algorithms' performance. Our CCGPMA achieves the best predictive performance in overall accuracy and AUC scores. Moreover, the non-linear competitors' KAAR and DL-based models achieve competitive results. However, GPC-MV, MA-GPC, and MA-GPCV offer the lowest classification scores. Regarding GPC-MV, the result is explained because GPC-MV corresponds to the most naive approach. After all, it considers homogeneity in the labelers' performance. Conversely, MA-GPC and MA-GPCV achieve a worse performance than GPC-MV; such behavior indicates that those methods do not properly represent the labelers' behavior. Regarding linear approaches, we highlight the outcome for MA-LFC, which achieves competitive performance compared to the non-linear competitors.

To summarize, we tested our approach in controlled scenarios by using two strategies. First, we simulate correlated annotators, whose performance is a function of the input space, named *SLFM-C*. The remaining strategy, named *Non-homogeneous accuracy*, simulates inconsistent annotators, *i.e.*, labelers, whose performance varies depending on the input features. Attained to the results (Table 5.2(a), and Table 5.2(b)), we note that our CCGPMA outperforms all its competitors considering both metrics AUC and the overall accuracy.

Finally, we test the *fully real datasets*, configuring the most challenging scenario. The input features and the labels from multiple experts come from real-world applications. Table 5.3 outlines the achieved AUC. First, we observe that for the voice data, G and R scales exhibit a similar AUC for all considered approaches; in fact, GPC-MV obtains a result comparable with the upper bound GPC-GOLD. The latter can be explained in that the annotators exhibit a suitable performance for these scales, *i.e.*, the provided labels are similar to the ground truth. On the other hand, a reduction in the predictive AUC is observed for scale B, which is a consequence of diminishing the labelers' performance compared to scales G and R, as demonstrated in [21]. Our approaches exhibit the best generalization performances for the three scales in the voice dataset. Remarkably, CGPMA-C and CCGPMA-C do not suffer significant changes in scale B. This is an outstanding outcome because it reflects that our method better represents the labelers' behavior against low-quality annotations. Finally, we review the AUC for the Music dataset. Achieved results show a low performance for the MA-GPC, even lower than their intuitive lower bound (GPC-MV). Notably, our CCGPMA-C reaches the best predictive AUC, comparable with the

intuitive upper bound.

Table 5.3 AUC classification results for the fully real datasets

Method	Voice			Music	Average
	G	R	B		
GPC-GOLD($M = 40$)	0.9481	0.9481	0.9481	0.9358	0.9450
GPC-GOLD($M = 80$)	0.9484	0.9484	0.9484	0.9178	0.9407
GPC-MV($M = 40$)	0.8942	0.9373	0.8001	0.8871	0.8797
GPC-MV($M = 80$)	0.9301	0.9377	0.7962	0.8897	0.8884
MA-LFC-C	0.9122	0.9130	0.8406	0.8599	0.8814
MA-DGRL	0.9127	0.9164	0.8259	0.8832	0.8845
MA-GPC	0.8660	0.8597	0.4489	0.8253	0.7500
MA-GPCV	0.9283	0.9208	0.8835	0.8677	0.9001
MA-DL-MW	0.8957	0.8966	0.8123	0.8567	0.8653
MA-DL-VW	0.8942	0.8929	0.8092	0.9167	0.8782
MA-DL-VW+B	0.9030	0.8937	0.8218	0.8573	0.8689
KAAR	0.9109	0.9351	0.8969	0.8896	0.9081
CGPMA-C($M = 40$)	**0.9324**	0.9406	0.8696	0.9025	0.9113
CGPMA-C($M = 80$)	**0.9324**	0.9417	0.8708	0.8987	0.9109
CCGPMA-C($M = 40$)	0.9318	**0.9422**	**0.9002**	0.9446	**0.9297**
CCGPMA-C($M = 80$)	0.9243	0.9383	0.8907	**0.9456**	0.9247

Bold: the highest performance excluding the GPC-GOLD bound.

5.4.2 Regression

We first perform a controlled experiment aiming to verify the capability of our CGPMA and CCGPMA to estimate the performance of inconsistent annotators as a function of the input space and take into account their dependencies. For this first experiment, we use the *fully synthetic* dataset described in Section 5.3.2. We simulate five labelers ($R = 5$) with different levels of expertise. To simulate the error variances based on the approach SLFM-R, we define $Q = 3$ functions $\hat{\mu}_q(\cdot)$, which are given as

$$\hat{\mu}_1(x) = 4.5\cos(2\pi x + 1.5\pi) - 3\sin(4.3\pi x + 0.3\pi) + 4\cos(7\pi x + 2.4\pi), \tag{5.34}$$

$$\hat{\mu}_2(x) = 4.5\cos(1.5\pi x + 0.5\pi) + 5\sin(3\pi x + 1.5\pi) - 4.5\cos(8\pi x + 0.25\pi), \tag{5.35}$$

$$\hat{\mu}_3(x) = 1, \tag{5.36}$$

where $x \in [0, 1]$. Besides, we define the following combination matrix $\hat{\boldsymbol{W}} \in \mathbb{R}^{Q\times R}$, where

$$\hat{\boldsymbol{W}} = \begin{bmatrix} -0.10 & 0.01 & -0.05 & 0.01 & -0.01 \\ 0.10 & -0.01 & 0.01 & -0.05 & 0.05 \\ -2.3 & -1.77 & 0.54 & 0.9 & 1.42 \end{bmatrix}, \tag{5.37}$$

holding elements $w_{l_r,q}$. Likewise, For the *Non-homogeneous error-variance* approach, we divide the input space into five regions and define the performance matrix $\boldsymbol{V}$ defined by equation (3.15).

Figure 5.3 shows the predictive performance of all methods in this first experiment. The results show two clear groups: those based on GPs (GPR-Av, MA-GPR, CGPMA-R, and CCGPMA-

Figure 5.3 Fully synthetic dataset results. We compare the prediction of our CCGPMA-R ($R^2 = 0.9438$), and CGPMA-R ($R^2 = 0.9280$) with the theoretical upper bound GPR-GOLD ($R^2 = 0.9843$) and lower bound GPR-Av ($R^2 = 0.8718$), and state-of-the-art approaches, MA-LFCR ($R^2 = -0.0245$), MA-GPR ($R^2 = 0.9208$), MA-DL-B ($R^2 = 0.7020$), MA-DL-S ($R^2 = 0.6559$), MA-DL-B+S ($R^2 = 0.5997$). Note that we provided the Gold Standard in dashed lines. The shaded region in GPR-Av, MA-GPR, CGPMA-R, and CCGPMA-R indicate the area enclosed by the mean plus or minus two standard deviations. We remark that there is no shaded region for MA-LFCR and DLMA since they do not provide information about the prediction uncertainty

R), which expose the best performance in terms of the R^2 score, and those based on other types of approaches (MA-LFCR, and MA-DL), whose performance is not satisfactory. The behavior of MA-LFCR is low since it only can deal with linear problems. Besides, concerning MA-DL and its three variations (S, B, and S+B), we note that this approach can deal with non-linear dynamics. However, MA-DL reaches a significantly low performance (even lower than the most naive approach, GPR-Av). To explain such an outcome, we remark that MA-DL introduces an additional layer, the "CrowdLayer," which allows the training of neural networks directly from the noisy labels of multiple annotators [25]. However, such a CrowdLayer properly codify the annotators' performance to guarantee a low computational cost [49]; therefore, MA-DL does not properly codify the annotators' behavior. On the other hand, among the GP-based methods, the proposed CCGPMA-R achieves the best performance in terms of R^2, followed closely by CGPMA-R and MA-GPR.

Besides, concerning the high performance of our CCGPMA-R (the best in terms of R^2 score), we hypothesize that such an outcome is a consequence of our method offering a better representation of the labelers' behavior when compared to its competitors. To empirically support the above hypothesis, Figure 5.4 shows the estimated error variances for this first experiment; here, we only consider the models that include these parameters in their formulations. As seen in Figure 5.4, MA-LFCR and MA-GPR offer the worst representation of the annotator's performance, which is expected because such models do not consider the relationship between the annotators and the input space. Conversely, CGPMA-R and CCGPMA-R outperform the models named previously. This outcome is a consequence that such two approaches compute the error-

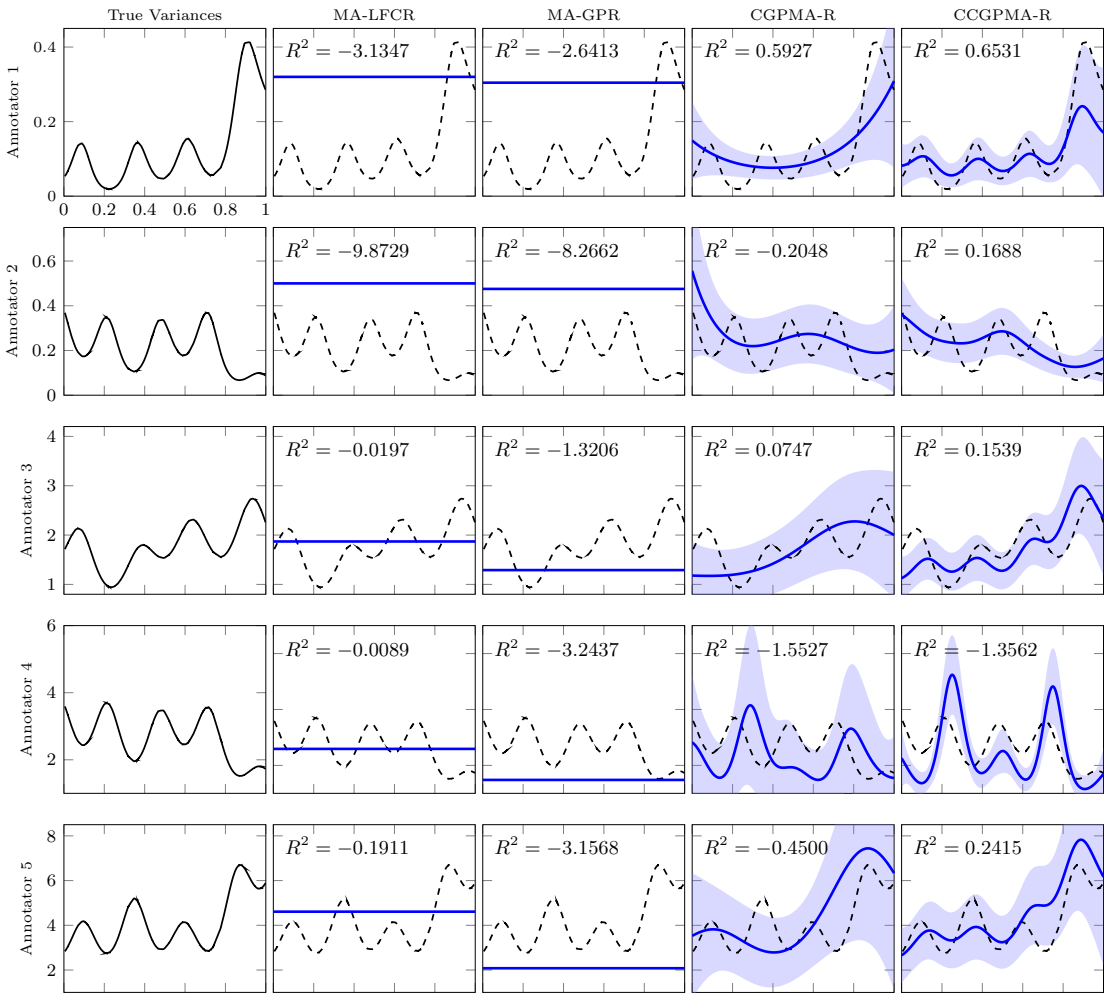

Figure 5.4 Estimated values of error variance for the five annotators in the *fully synthetic* experiment. In the first column, from top to bottom, we expose the error variances used to simulate the labels from each annotator. Furthermore, the subsequent columns from top to bottom present the estimation of such error variances performed by state-of-the-art models that include these parameters in their formulation; moreover, the true error variances are provided in dashed lines. The shaded region in CGPMA-R and CCGPMA-R indicates the area enclosed by the mean plus or minus two standard deviations. We remark that there is no shaded region for MA-LFCR and MA-GPR since these approaches perform a fixed-point estimation for the annotators' parameters. Finally, we remark that the R^2 score between the true and estimated error variances is provided

variance as a function of the input features, allowing for a better codification of the labelers' behavior. Besides, by visually inspecting and analyzing the R^2 scores, CCGPMA-R performs better than CGPMA-R because the former properly codes the annotators' interdependencies [10]. Finally, although our CCGPMA-R achieves the best representation of the annotators' performance, the result for Annotator 4 exhibits a lower performance in terms of R^2 score compared to the other labelers. The quasi-periodic behavior causes such an outcome in the error variances for those labelers, which our approach cannot capture because we are using an RBF-based kernel.

Second, Table 5.4 (a) shows the results of the *semi-synthetic datasets*. On average, our CCGPMA-R exhibits the best generalization performance regarding the R^2 score. On the other hand, regarding its GPs-based competitors (GPR-Av, MA-GPR, and CGPMA-R), we first note

Table 5.4 Semi-synthetic regression results

(a) SLFM-R

Method		Auto	Bike	Concrete	Housing	Yacht	CT	Average
GPC-GOLD($M=40$)	R^2	0.8604 ± 0.0271	0.5529 ± 0.0065	0.8037 ± 0.0254	0.8235 ± 0.0419	0.8354 ± 0.0412	0.8569 ± 0.0055	0.7888
	Pear	0.9359 ± 0.0132	0.7440 ± 0.0046	0.8997 ± 0.0142	0.9152 ± 0.0209	0.9350 ± 0.0110	0.9268 ± 0.0030	0.8928
GPC-GOLD($M=80$)	R^2	0.8612 ± 0.0279	0.5603 ± 0.0063	0.8271 ± 0.0230	0.8275 ± 0.0399	0.8240 ± 0.0339	0.8648 ± 0.0047	0.7942
	Pear	0.9361 ± 0.0139	0.7490 ± 0.0042	0.9124 ± 0.0126	0.9174 ± 0.0198	0.9281 ± 0.0141	0.9309 ± 0.0025	0.8956
GPR-MV($M=40$)	R^2	0.8425 ± 0.0286	0.5280 ± 0.0100	0.7589 ± 0.0279	0.7834 ± 0.0463	0.7588 ± 0.0498	0.8070 ± 0.0130	0.7464
	Pear	0.9260 ± 0.0155	0.7289 ± 0.0073	0.8765 ± 0.0168	0.8966 ± 0.0265	0.8913 ± 0.0162	0.9067 ± 0.0058	0.8710
GPR-MV($M=80$)	R^2	0.8406 ± 0.0304	0.5397 ± 0.0085	0.7765 ± 0.0274	0.7903 ± 0.0451	0.7676 ± 0.0535	0.8167 ± 0.0089	0.7552
	Pear	0.9246 ± 0.0168	0.7378 ± 0.0056	0.8852 ± 0.0162	0.8995 ± 0.0253	0.8964 ± 0.0184	0.9117 ± 0.0030	0.8759
MA-LFCR	R^2	0.7973 ± 0.0218	0.3385 ± 0.0051	0.6064 ± 0.0384	0.7122 ± 0.0509	0.6403 ± 0.0186	0.8400 ± 0.0014	0.6558
	Pear	0.8953 ± 0.0116	0.5892 ± 0.0058	0.7823 ± 0.0230	0.8497 ± 0.0311	0.8091 ± 0.0137	0.9172 ± 0.0008	0.8071
MA-GPR	R^2	0.8456 ± 0.0281	0.4448 ± 0.0187	0.7769 ± 0.0367	0.7685 ± 0.0632	0.7842 ± 0.1027	0.0105 ± 0.0045	0.6051
	Pear	0.9227 ± 0.0148	0.6694 ± 0.0140	0.8834 ± 0.0209	0.8809 ± 0.0352	0.9007 ± 0.0427	0.1410 ± 0.0072	0.7330
MA-DL-B	R^2	0.7766 ± 0.0253	$\mathbf{0.5854 \pm 0.0107}$	0.2319 ± 0.0328	0.5317 ± 0.1005	0.2089 ± 0.0783	0.6903 ± 0.2689	0.5041
	Pear	0.8951 ± 0.0190	0.7712 ± 0.0047	0.5168 ± 0.0327	0.7555 ± 0.0619	0.5123 ± 0.0892	0.9698 ± 0.0138	0.7368
MA-DL-S	R^2	0.7761 ± 0.0279	0.5828 ± 0.0149	0.2363 ± 0.0252	0.5352 ± 0.0948	0.1822 ± 0.0985	$\mathbf{0.8418 \pm 0.2368}$	0.5257
	Pear	0.8977 ± 0.0160	$\mathbf{0.7736 \pm 0.0060}*$	0.5175 ± 0.0322	0.7540 ± 0.0607	0.4883 ± 0.1189	$\mathbf{0.9584 \pm 0.0161}*$	0.7316
MA-DL-B+S	R^2	0.7717 ± 0.0239	0.5816 ± 0.0181	0.2369 ± 0.0322	0.5330 ± 0.0850	0.1974 ± 0.0895	0.5517 ± 0.2316	0.4787
	Pear	0.8936 ± 0.0182	0.7727 ± 0.0071	0.5172 ± 0.0429	0.7537 ± 0.0580	0.5048 ± 0.0969	0.9632 ± 0.0157	0.7342
CGPMA-R($M=40$)	R^2	0.8476 ± 0.0229	0.5464 ± 0.0069	0.8169 ± 0.0231	0.7244 ± 0.2973	0.8049 ± 0.0482	0.8236 ± 0.0132	0.7606
	Pear	0.9280 ± 0.0109	0.7395 ± 0.0047	0.9063 ± 0.0126	0.8234 ± 0.2705	0.9183 ± 0.0235	0.9117 ± 0.0066	0.8712
CGPMA-R($M=80$)	R^2	0.8342 ± 0.0217	0.5560 ± 0.0074	0.8190 ± 0.0254	0.7259 ± 0.3018	0.7928 ± 0.0884	0.8371 ± 0.0104	0.7608
	Pear	0.9212 ± 0.0119	0.7459 ± 0.0051	0.9078 ± 0.0137	0.8093 ± 0.3051	0.9206 ± 0.0348	0.9188 ± 0.0046	0.8706
CCGPMA-R($M=40$)	R^2	$\mathbf{0.8558 \pm 0.0248}$	0.5284 ± 0.0117	0.7976 ± 0.0270	0.8169 ± 0.0468	0.8409 ± 0.0548	0.8219 ± 0.0062	0.7769
	Pear	$\mathbf{0.9323 \pm 0.0115}$	0.7345 ± 0.0050	0.8960 ± 0.0153	0.9129 ± 0.0263	0.9454 ± 0.0109	0.9123 ± 0.0017	0.8889
CCGPMA-R($M=80$)	R^2	0.8534 ± 0.0243	0.5467 ± 0.0069	$\mathbf{0.8220 \pm 0.0259}$	$\mathbf{0.8215 \pm 0.0466}$	$\mathbf{0.8691 \pm 0.0473}$	0.8252 ± 0.0083	**0.7897**
	Pear	0.9305 ± 0.0109	0.7443 ± 0.0039	$\mathbf{0.9092 \pm 0.0140}*$	$\mathbf{0.9150 \pm 0.0255}*$	0.9583 ± 0.0102	$\mathbf{0.9138 \pm 0.0017}$	**0.8952**$*$

(b) Non-homogeneous error-variance

Method		Auto	Bike	Concrete	Housing	Yacht	CT	Average
GPC-GOLD($M=40$)	R^2	0.8604 ± 0.0271	0.5529 ± 0.0065	0.8037 ± 0.0254	0.8235 ± 0.0419	0.8354 ± 0.0412	0.8569 ± 0.0055	0.7888
	Pear	0.9359 ± 0.0132	0.7440 ± 0.0046	0.8997 ± 0.0142	0.9152 ± 0.0209	0.9350 ± 0.0110	0.9268 ± 0.0030	0.8928
GPC-GOLD($M=80$)	R^2	0.8612 ± 0.0279	0.5603 ± 0.0063	0.8271 ± 0.0230	0.8275 ± 0.0399	0.8240 ± 0.0339	0.8648 ± 0.0047	0.7942
	Pear	0.9361 ± 0.0139	0.7490 ± 0.0042	0.9124 ± 0.0126	0.9174 ± 0.0198	0.9281 ± 0.0141	0.9309 ± 0.0025	0.8956
GPR-MV($M=40$)	R^2	0.8388 ± 0.0373	0.5348 ± 0.0065	0.7648 ± 0.0262	0.7795 ± 0.0508	0.7752 ± 0.0579	0.8169 ± 0.0103	0.7517
	Pear	0.9277 ± 0.0154	0.7335 ± 0.0055	0.8784 ± 0.0151	0.8921 ± 0.0280	0.9030 ± 0.0152	0.9109 ± 0.0043	0.8743
GPR-MV($M=80$)	R^2	0.8402 ± 0.0396	0.5471 ± 0.0056	0.7819 ± 0.0270	0.7892 ± 0.0468	0.7792 ± 0.0616	0.8268 ± 0.0061	0.7607
	Pear	0.9281 ± 0.0170	0.7421 ± 0.0045	0.8874 ± 0.0154	0.8968 ± 0.0259	0.9061 ± 0.0166	0.9153 ± 0.0030	0.8793
MA-LFCR	R^2	0.7909 ± 0.0210	0.3868 ± 0.0065	0.6050 ± 0.0333	0.7045 ± 0.0610	0.6230 ± 0.0290	0.8605 ± 0.0014	0.6618
	Pear	0.8992 ± 0.0098	0.6223 ± 0.0051	0.7817 ± 0.0198	0.8512 ± 0.0284	0.8024 ± 0.0119	0.9277 ± 0.0007	0.8141
MA-GPR	R^2	$\mathbf{0.8597 \pm 0.0355}$	0.4408 ± 0.0169	0.7921 ± 0.0269	0.7446 ± 0.0624	0.7800 ± 0.0565	0.0116 ± 0.0034	0.6048
	Pear	0.9300 ± 0.0180	0.6661 ± 0.0125	0.8915 ± 0.0153	0.8661 ± 0.0348	0.8899 ± 0.0309	0.1415 ± 0.0068	0.7309
MA-DL-B	R^2	0.7681 ± 0.0297	$\mathbf{0.5996 \pm 0.0105}$	0.2543 ± 0.0267	0.5264 ± 0.0973	0.1843 ± 0.0820	0.7612 ± 0.2679	0.5156
	Pear	0.8923 ± 0.0196	0.7804 ± 0.0055	0.5395 ± 0.0387	0.7466 ± 0.0572	0.4963 ± 0.0970	0.9806 ± 0.0061	0.7393
MA-DL-S	R^2	0.7687 ± 0.0316	0.5944 ± 0.0097	0.2495 ± 0.0332	0.5279 ± 0.1031	0.1809 ± 0.0963	0.8231 ± 0.3349	0.5241
	Pear	0.8919 ± 0.0171	$\mathbf{0.7830 \pm 0.0051}$	0.5312 ± 0.0407	0.7484 ± 0.0604	0.4840 ± 0.1281	0.9801 ± 0.0091	0.7364
MA-DL-B+S	R^2	0.7765 ± 0.0315	0.5909 ± 0.0113	0.2446 ± 0.0379	0.5454 ± 0.0933	0.1889 ± 0.0813	0.6294 ± 0.2498	0.4960
	Pear	0.8954 ± 0.0168	0.7813 ± 0.0051	0.5251 ± 0.0482	0.7598 ± 0.0571	0.4946 ± 0.1163	0.9747 ± 0.0122	0.7385
CGPMA-R($M=40$)	R^2	0.8570 ± 0.0335	0.5462 ± 0.0062	0.8154 ± 0.0265	0.8260 ± 0.0503	0.8025 ± 0.0748	0.8561 ± 0.0062	0.7839
	Pear	$\mathbf{0.9338 \pm 0.0154}*$	0.7399 ± 0.0044	0.9055 ± 0.0148	0.9166 ± 0.0253	0.9177 ± 0.0417	0.9264 ± 0.0029	0.8900
CGPMA-R($M=80$)	R^2	0.8137 ± 0.0667	0.5544 ± 0.0069	$\mathbf{0.8250 \pm 0.0260}$	$\mathbf{0.8302 \pm 0.0479}$	0.7458 ± 0.1670	$\mathbf{0.8622 \pm 0.0049}$	0.7719
	Pear	0.9133 ± 0.0298	0.7456 ± 0.0049	$\mathbf{0.9107 \pm 0.0143}*$	$\mathbf{0.9185 \pm 0.0243}*$	0.9064 ± 0.0748	$\mathbf{0.9296 \pm 0.0024}*$	0.8873
CCGPMA-R($M=40$)	R^2	0.8567 ± 0.0342	0.5265 ± 0.0068	0.7980 ± 0.0264	0.8172 ± 0.0548	0.8442 ± 0.0738	0.8006 ± 0.0093	0.7739
	Pear	0.9342 ± 0.0156	0.7340 ± 0.0033	0.8960 ± 0.0151	0.9116 ± 0.0282	0.9566 ± 0.0064	0.9035 ± 0.0024	0.8893
CCGPMA-R($M=80$)	R^2	0.8571 ± 0.0326	0.5428 ± 0.0073	0.8202 ± 0.0257	0.8288 ± 0.0515	$\mathbf{0.8687 \pm 0.0587}$	0.8403 ± 0.0099	**0.7930**
	Pear	0.9335 ± 0.0160	0.7429 ± 0.0045	0.9082 ± 0.0144	0.9183 ± 0.0258	$\mathbf{0.9653 \pm 0.0136}*$	0.9205 ± 0.0040	**0.8981**$*$

Bold: the highest R^2 excluding the upper bound (target) classifier GPR-GOLD. Marked with $*$: the highest Pearson coefficient (Pear) except the upper bound.

that the performance of CGPMA-R exhibits a similar (but lower) performance than CCGPMA-R. The above is a consequence conversely to CGPMA-R; our CCGPMA-R models the annotators' interdependencies. Secondly, the lower bound GPR-Av exhibits a significantly worse prediction than our approaches. We remark on MA-GPR's behavior, which is lowest compared to its GPs-based competitors, even far worse than GPR-Av. The key to this outcome lies in the MA-GPR

formulation; it models the annotators' behavior by assuming that their performance does not depend on the input features and considering that the labelers make their decisions independently, which does fit the process that we use to simulate the labels. Next, we analyze the results concerning the linear model MA-LFR; attained the results, we note that this approach's prediction capacity is far lower than ours. The above outcome suggests that there may exist a non-linear structure in most databases. However, we highlight a particular result for the dataset CT, where MA-LFCR exhibits the best performance defeating all its competitors based on non-linear models. The CT dataset may have a linear structure from the above. To confirm this supposition, we perform an additional experiment over CT by training a regression scheme based on LR with the actual labels (we follow the same scheme as for GPR-GOLD). We obtained an R^2 score equal to 0.8541 (on average), which is close to the results obtained by GPR-GOLD. Thus, we can elucidate that there exists a linear structure in the dataset CT. Finally, we analyze the results for the DL-based models. Similar to the experiments over *fully synthetic datasets*, we note a considerably low prediction capacity; they are even defeated by the linear model MA-LFR. Again, we attribute this behavior to the fact that the CrowdLayer (used to manage the data from multiple annotators) does not offer a suitable codification of the labelers' behavior. Nevertheless, considering the above, we observe a remarkable result in the Bike dataset, where the DL-based approaches offer the best performance, even defeating the supposed upper-bound GPR-GOLD. To explain that, it is necessary to analyze the meaning of the target variable in such a dataset. Regarding the description of this dataset,[3] the target variables indicate the count of total rental bikes, including casual and registered, in a day. The above suggests that there may exist a quasi-periodic structure in the dataset, which the GPR-GOLD cannot capture since it uses a non-periodic kernel (RBF). To support our suppositions, an additional experiment was performed over this dataset by training the model GPR-GOLD with the following kernel:

$$\kappa(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) = s^2 \exp\left[-\frac{1}{2}\sum_{p=1}^{P}\left(\frac{\sin(\frac{\pi}{T_p}(x_{p,n} - x_{p,n'}))}{l_p}\right)^2\right], \tag{5.38}$$

where $s \in \mathbb{R}$ is the variance parameter, $l_p \in (\mathbb{R}^+)$ is the length-scale parameter for the p-th dimension, and $T_p \in (\mathbb{R}^+)$ is the period for the p-th dimension. Therefore, we obtain an R^2 score equal to 0.5952 (on average), which is greater than the DL-based approaches, indicating a quasi-periodic structure in the Bike dataset, as we had supposed.

On the other hand, Table 5.4(b) presents the results concerning the simulation method "Non-homogeneous error-variance". We highlight that our CCGPMA exhibits the best regression performance in both metrics R^2 score and the Pearson coefficient. Now, analyzing the behavior of GPs-based competitors, we notice that GPR-Av exhibits the best performance compared to MA-GPR. Our explanation for such an outcome lies in the formulation of MA-GPR since it models the

[3]Such description can be found in https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

annotators' behavior without considering relationships between the input space and the annotators' performance. Regarding the linear model, MA-LFR, we note that its prediction performance is considerably lower than our method. The above indicates the presence of a non-linear structure in most considered datasets. Finally, analyzing the DL-based methods, we remark that their performance is the lowest concerning the average of the R^2 score. Our explanation for this result is that the Crowdlayer used for the DL models to manage multi-labeler data does not offer a suitable representation of the annotators' behavior.

Finally, we use *fully real datasets*, which present the most challenging scenario, where both the input samples and the labels come from real-world applications. Table 5.5 outlines the achieved performances. We remark that our CCGPMA-R with $M = 80$ obtains the best generalization performance in terms of the R^2 score. Further, as theoretically expected, its performance lies between that of GPR-GOLD and GP-Av. Moreover, regarding the GPs-based competitors (MA-GPR and CGPMA-R), we note that our CGPMA-R is lower than CCGPMA-R. On the other hand, MA-GPR exhibits the worst prediction capability with an R^2 close to zero. We suppose the above is a symptom of overfitting, which can be confirmed due to the training R^2 score for MA-GPR is 0.4731, which is comparable with GPR-GOLD. Conversely, the linear approach MA-LFCR exhibits the second-lowest performance and performs worse than the theoretical lower bound GP-Av indicates a non-linear structure in the Music dataset. Finally, analyzing the results from the deep learning approaches, we note that the variation MA-DL-B+S exhibits a similar performance compared to our CGPMA-R; however, it is slightly lower than our CCGPMA-R. We highlight that despite deep learning capacities, our approach CCGPMA-R offers a better representation of annotators' behavior, unlike the deep learning techniques, which measure such performance using a single parameter.

Table 5.5 Regression results regarding R^2 score over the *fully real dataset*

Method	Music
GPR-GOLD($M = 40$)	0.4704
GPR-GOLD($M = 80$)	0.4889
GPR-Av($M = 40$)	0.2572
GPR-Av($M = 80$)	0.2744
MA-LFCR	0.1404
MA-GPR	0.0090
MA-DL-B	0.2339
MA-DL-S	0.2934
MA-DL-B+S	0.3519
CGPMA-R($M = 40$)	0.3345
CGPMA-R($M = 80$)	0.3531
CCGPMA-R($M = 40$)	0.3337
CCGPMA-R($M = 80$)	$\mathbf{0.3872}$

Bold: the highest R^2 excluding the upper bound GPR-GOLD.

5.5 Summary

We propose a novel Gaussian Process-based approach to deal with Multiple Annotators scenarios, termed Correlated Chain Gaussian Process for Multiple Annotators (CCGPMA). Our method is built as an extension of the chained GP [59], introducing a semi-parametric latent factor model-(SLFM) to exploit correlations between the GP latent functions that model the parameters of a given likelihood function. To the best of our knowledge, CCGPMA is the first attempt to build a probabilistic framework that codes the annotators' expertise as a function of the input data and exploits the correlations among the labelers' answers. Besides, we highlight that our approach can be used with different likelihoods, which allows us to deal with both categorical data (classification) and real-valued (regression). We tested our approach for classification and regression tasks using different scenarios concerning the provided annotations: synthetic, semi-synthetic, and real-world experts. According to the results, our CCGPMA can achieve robust predictive properties for the studied datasets, outperforming state-of-the-art methods.
On the other hand, as observed in Section 5.4.2, kernel selection plays a key role in the performance of our CCGPMA. Hence, An improper kernel selection leads to unsatisfactory performance.

Chapter 6 Final Remarks

6.1 Conclusions

This book presented several specific methodologies based on frequentist and Bayesian models to face supervised learning problems with multiple annotators. In that sense, five strategies were proposed to code the annotators' behavior, considering the relationship between the input features, the labelers' performance and the dependencies among their outputs. The introduced approaches naturally lead to data-dependent processing tuned to the particular sample constraints and the considered learning scenario, including regression and classification (binary and multiclass) tasks. The introduced framework is tested in both controlled scenarios, where the input samples and the labels from multiple annotators come from real-world problems. Overall, attained results demonstrated that proposed approaches code the labelers' performance taking into account dependencies among the annotators and considering the labelers' performance as a function of the input data, favoring the learning performance in terms of task accuracy and data interpretability in comparison to state-of-the-art methods. Following, the main concluding remarks regarding each provided representation strategy are described:

- A kernel-alignment-based approach that allows coding the annotators' performance in the absence of the ground truth. The introduced strategy, termed KAAR, can compute each provided labeler's relevance through a CKA-based averaged matching between the given labels and the input data features; it also considers similarities between all annotators' labels. Hence, KAAR relaxes the assumption of independence between the labelers, which allows coding some biases and tendencies between the annotators' opinions. Then, a convex combination of supervised learning algorithms is carried out by adopting the multiple annotator performances coded in the KAAR-based relevance analysis. For comparison, KAAR is tested in synthetic and real-world datasets for classification and regression settings. Results show that KAAR can deal with binary classification, multiclass classification, and regression problems with multiple annotators.
- A localized kernel-alignment-based annotator relevance approach, named LKAAR, supports binary, multi-class classification, and regression problems in the presence of multiple annotators, which configures an extension of KAAR aiming to code the relationship between the input features and the labelers' behavior. Accordingly, our LKAAR computes the relevance of each provided expert through a centered kernel alignment-based matching between the annotator labels and the input features, taking into account dependencies among the annotators and considering the labelers' performance as a function of the input data. Like KAAR, the LKAAR-based relevance analysis is used as weights in a combi-

nation of classifiers/regressors (one for labeler), which are used to solve the supervised learning problem from multiple annotators. We tested our approach in synthetic, semi-synthetic, and real-world datasets. The results show that the proposed method deals with binary, multi-class classification, and regression problems, where the ground truth is not available. In most cases, our LKAAR achieves competitive or even better results when compared to different state-of-the-art classification and regression approaches.

- A regularized chained Deep neural network (RCDNN) to support binary and multiclass classification. RCDNN is built based on the concepts of the chained Gaussian processes [59], where the parameters of a given multi-labeler likelihood are linked to the output neurons of a deep neural network, where some of those parameters are related to the annotators' performance. Accordingly, RCDNN codes such performance as a function of the input space $\mathcal{X}$ and the annotators' interdependencies. Besides, l1, l2, and Monte-Carlo Dropout regularization strategies are coupled within our RCDNN architecture and predictor to contract the overfitting challenge of deep models. The proposal is tested over binary and multiclass classification settings using different scenarios concerning the provided annotations: synthetic, semi-synthetic, and real-world experts. According to the results, RCDNN achieves robust predictive properties for the studied datasets, outperforming state-of-the-art methods while providing an estimation of each labeler's reliability and the dependencies among annotators.
- A novel Gaussian Process-based approach to deal with Multiple labelers scenarios termed Correlated Chain Gaussian Process for Multiple Annotators (CCGPMA). Our method is built as an extension of the chained GP [59], introducing a semi-parametric latent factor model-(SLFM) to exploit correlations between the GP latent functions that model the parameters of a given likelihood function. Hence, our CCGPMA jointly solves the supervised learning problem and estimates the annotators' behavior. Notably, CCGPMA allows codifying the labelers' performance as a function of the input space while capturing correlations between them. We highlight that our approach can be used with different likelihood functions, which allows us to deal with categorical data (classification) and real-valued (regression). We tested our approach for classification and regression tasks using different scenarios concerning the provided annotations: synthetic, semi-synthetic, and real-world experts. According to the results, our CCGPMA can achieve robust predictive properties for the studied datasets, outperforming state-of-the-art methods.

Then, for comparison purposes, in Table 6.1, we expose the results of our approaches and the state-of-the-art methods in Table 5.1 over three semi-synthetic datasets (Ionosphere, Western, and Segmentation), where the labels were simulated by following the procedure Section 5.3.1, which is the most challenging since it simulates correlated labelers, where their performance depends on the input features. We show the non-parametric Friedman test results to establish the statistical significance of such results. The null hypothesis settles that all algorithms perform

equally [82]. Also, we fix the significance threshold as $p < 0.05$.

From Table 6.1, we note that analyzing both the average AUC and the average ranking, the approaches proposed in this research are competitive (KAAR, LKAAR) or even outperform (RCDNN, CGPMA, and CCGPMA) the considered state-of-the-art approaches. Besides focusing on our approaches, we notice two groups; the first includes the methods CCGPMA, CGPMA, and RCDNN (exhibit the best performance) that jointly estimates the annotators' performance and the supervised function f. On the other hand, the second group is conformed by the approaches that perform such estimation independently, KAAR and LKAAR (the lowest performance). We highlight that the first group offers better prediction performance regarding the AUC score. The above is explained in the sense that, as it has been mentioned in different studies [1, 16, 19, 49], methods that jointly build the supervised learning algorithm and the annotators' performance achieve superior performance since the interaction between such factors (labelers' behavior and the supervised learning model) provides critical information to puzzle out the actual labels. We apply the Friedman test to verify the significance of the results in Table 6.1. As seen, we obtain a Chi-square of 26.84 with $p - value = 0.0084$. Thus, we have enough statistical evidence to determine that our CCGPMA performs better than state-of-the-art competitors.

Table 6.1 AUC classification results for the semi-synthetic and fully real datasets

Method	Ionosphere	Western	Segmentation	Voice G	Voice R	Voice B	Music	Average AUC	Average ranking
GPC-GOLD	0.9513	0.9250	0.9781	0.9484	0.9484	0.9484	0.9358	0.9479	−−
GPC-MV	0.7779	0.8658	0.9749	0.9301	0.9377	0.8001	0.8897	0.8823	6.1428
MA-LFC-C	0.7358	0.8400	0.9892	0.9122	0.9130	0.8406	0.8599	0.8701	8.0000
MA-DGRL	0.6453	0.8143	0.9897	0.9127	0.9164	0.8259	0.8832	0.8553	8.7142
MA-GPC	0.6631	0.8671	0.9734	0.8660	0.8597	0.4489	0.8253	0.7862	11.142
MA-GPCV	0.6238	0.8451	0.9924	0.9283	0.9208	0.8835	0.8677	0.8659	7.4285
MA-DL-MW	0.7535	0.9092	0.9950	0.8957	0.8966	0.8123	0.8567	0.8741	7.8571
MA-DL-VW	0.6987	0.9173	**0.9972**	0.8942	0.8929	0.8092	0.9167	0.8751	7.4285
MA-DL-VW+B	0.7196	0.9164	**0.9972**	0.9030	0.8937	0.8218	0.8573	0.8727	7.5714
KAAR	0.7046	0.8588	0.9217	0.9109	0.9351	0.8969	0.8896	0.8739	7.8571
LKAAR	0.7121	0.8359	0.9355	0.9206	0.9360	0.8979	0.8998	0.8768	7.1428
RCDNN	0.6818	0.8520	0.9634	0.9224	0.9419	**0.9257**	0.9329	0.8885	5.7142
CGPMA-C	0.8615	0.9185	0.9679	**0.9324**	0.9417	0.8708	0.9025	0.9136	4.0000
CCGPMA-C	**0.9023**	**0.9307**	0.9774	0.9318	**0.9422**	0.9002	**0.9456**	**0.9328**	**2.1428**

Bold: the highest performance excluding the GPC-GOLD bound. The last column presents the average ranking for the AUC score and the best average ranking in bold. The Friedman test returns a Chi-square value of 26.84 ($p - value = 0.0081$).

Finally, we intend to analyze the impact of malicious annotators on our approaches' performance and compare them with some state-of-the-art methods. Similar to Section 4.4, we employ the Pima dataset with 768 instances; thus, we use 538 samples for training and the remaining 230 for testing. We create synthetic labels from 5 annotators generated from the Biased coin (Non-homogeneous) procedure (Section 3.3.1 and equation (4.10)). According to Figure 6.1 (left), from the 5 labelers (red dots), two are categorized as suitable labelers, one as Spammers, and the remaining as Malicious. Then, we simulate 20 additional annotators to analyze the behavior

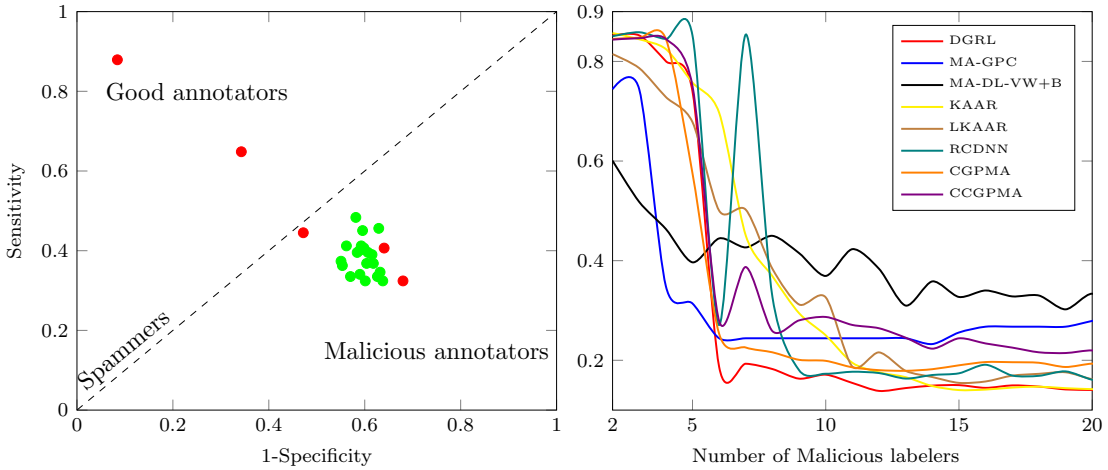

Figure 6.1 Malicious annotators. On the left, we show the Receiver's Operating Characteristic (ROC) plot for the simulated annotators. Red dots indicate the basis annotators. Green dots show malicious annotators. On the right, we present the performance (AUC) of DGRL, MA-GPC, MA-DL-VW+B, KAAR, LKAAR, RCDNN, CGPMA, and CCGPMA as a function of the number of malicious annotators

of our approaches in scenarios where the number of malicious annotators increases. The labels from malicious annotators are generated as as follows (see green dots on the left of Figure 6.1): a random number $\alpha_n^{(r)}$ is sampled from a Bernoulli distribution with parameter $p_r = 0.6$; then if $\alpha_n^{(r)} = 0$, $y_n^{(r)} = y_n$, and $y_n^{(r)} = \tilde{y}_n$ otherwise.

From the results in Figure 6.1 (right), we note that our approaches are significantly affected when the number of malicious annotators is great than 5 malicious annotators, which suggests the presence of a critical point is presented when the percentages of good, spammers, and malicious labelers are respectively 25%, 12.5%, 62.5%. Accordingly, for this experiment, we can affirm that our approach can deal with malicious labelers if the percentage of them is below 62.5%. On the other hand, regarding the state-of-the-art competitors MA-DGRL, MA-GPC, and MA-DL-VW+B, we notice that all of them (except MA-DGRL) are more susceptible to malicious labelers; in fact, their AUC is affected whit three or more malicious annotators. Conversely, we highlight the performance of MA-DGRL, which is very close to our methods. To explain such an outcome, we remark that MA-GPC and MA-DL-VW+B estimate the annotators' performance as an average of some parameters. However, MA-DGRL estimates such performance for every region in the input space, which makes it more robust to the presence of malicious annotators.

6.2 Future Work

We have presented a framework based on frequentist and Bayesian models aiming to face supervised learning problems with data from multiple annotators. However, from the theoretical and experimental results, many issues can still be addressed to improve learning performance. In particular, the following remarks could be of interest to future work approaches:

- KAAR and LKAAR independently compute the annotators' behavior and the supervised

learning function f. Such a function f is estimated using a convex combination of R classifiers/regressors, which could be inconvenient for a large number of labelers. In that sense, an interesting extension of such works could be to design a model that jointly relates the supervised learning function and the annotators' behavior based on LKAAR or KAAR.

- Concerning RCDNN, extending RCDNN for regression tasks is an exciting research line, *i.e.*, based on the model introduced in [22]. Next, we plan to use other deep structures, *i.e.*, Convolutional and Recurrent layers and different activation functions, to apply our approach to more complex tasks such as computer vision or natural language processing.
- Moreover, regarding the RCDNN model, we recall that it only was proposed to deal with classification (binary and multi-class) tasks. Accordingly, we plan to extend such an approach to real-valued data to deal with interesting applications such as text regression [25].
- Regarding CCGPMA, we can use convolution processes [83] instead of the SLFM, aiming to represent the labelers' correlations better. Also, our approach can be extended for multi-task learning in the context of multiple annotators [40]. Accordingly, the likelihood of multi-task learning for multiple annotators can be given as follows:

$$p(\boldsymbol{Y}_1,\dots\boldsymbol{Y}_E|\boldsymbol{\theta}_1,\dots\boldsymbol{\theta}_E)=\prod_{e=1}^{E}\prod_{n=1}^{N}\prod_{r_e\in R_n^e} p(y_{r_e,n}^e|\theta_1(\boldsymbol{x}_n),\dots,\theta_{J_e}(\boldsymbol{x}_n)), \tag{6.1}$$

$\boldsymbol{Y}_e\in\mathbb{R}^{N\times R_e}$ is a matrix containing the labels $y_{r_e,n}^e$ given by R_e annotators for the e-th task. Further, $\boldsymbol{\theta}_e=[\boldsymbol{\theta}_1^e,\dots,\boldsymbol{\theta}_{J_e}^e]^\top\in\mathbb{R}^{NJ}$, and $\boldsymbol{\theta}_j^e=[\theta_j^e(\boldsymbol{x}_1),\dots,\theta_j^e(\boldsymbol{x}_N)]^\top\in\mathbb{R}^N$ is the j-th parameter corresponding to the e-th task.

- We recognize an increasing interest in multi-labeler approaches from the Bioinformatics community, particularly in applications related to computer-aided diagnosis–(CAD) systems [1, 36], such as predicting malignancy of pulmonary nodules, mitosis detection in breast cancer [3] and the assessment of voice quality [21]. In this type of system, the medical diagnosis (labels) represents the health risk levels, which are inherently ordered. Accordingly, an ordinal regression problem is configured [84], which cannot be tackled using multi-class classification or regression methods. In that sense, it is interesting to extend our approaches to deal with ordinal data aiming to contribute to the Bioinformatics area.
- Following the previous idea, we remark that in this work, we used a dataset termed "Voice," which comprises the evaluation of voice records following the perceptual scale GRBAS. Such a dataset configures an ordinal regression problem; however, we convert it into a binary classification problem for validation purposes (gold standard's availability). In that sense, it is interesting to apply our approaches to deal with the Voice dataset and build a system for the automatic assessment of voice quality based on the GRBAS scale in the presence of multiple annotators.
- Finally, this book is focused on training of SL algorithms from noisy labels adopting the *learning from crowds* point of view. However, there are other perspectives to cast the

noisy labels drawback. Specifically, we recognize the *self-learning* approaches [85], which correspond to a pre-training step within a DNN architecture to correct possibly noisy or missing label imputation. *Self-learning* approaches have been applied in computer vision [86] and natural language processing [87]. Thereby, we establish that developing a model that combines the concepts of *self-learning* and *learning from crowds* to face SL scenarios with noisy labels is interesting.

6.3 Repositories

Model	Github repository
KAAR	https://github.com/juliangilg/KAAR
LKAAR	https://github.com/juliangilg/LKAAR
RCDNN	https://github.com/juliangilg/RCDNN_MA
CCGPMA	https://github.com/juliangilg/CCGPMA

References

[1] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy, "Learning from crowds," *J. Speech Lang. Hear. Res.*, vol. 11, no. Apr, pp. 1297–1322, 2010.

[2] P. Welinder, S. Branson, P. Perona, and S. J. Belongie, "The multidimensional wisdom of crowds," in *Advances in neural information processing systems*, 2010, pp. 2424–2432.

[3] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab, "Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1313–1321, 2016.

[4] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: a survey," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 543–576, 2016.

[5] H.-E. Sung, C.-K. Chen, H. Xiao, and S.-D. Lin, "A classification model for diverse and noisy labelers," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2017, pp. 58–69.

[6] H. D. Vargas-Cardona, Á. A. Orozco, A. M. Álvarez, and M. A. Álvarez, "Tensor decomposition processes for interpolation of diffusion magnetic resonance imaging," *Expert Systems with Applications*, vol. 118, pp. 92–108, 2019.

[7] D. A. Jimenez, H. F. García, A. M. Álvarez, Á. A. Orozco, and G. Holguín, "A kernelized morphable model for 3d brain tumor analysis," in *International Conference Image Analysis and Recognition*. Springer, 2018, pp. 529–537.

[8] C. Jimenez, D. Diaz, D. Salazar, A. Alvarez, A. Orozco, and O. Henao, "Nerve structure segmentation from ultrasound images using random under-sampling and an svm classifier," in *International Conference Image Analysis and Recognition*. Springer, 2018, pp. 571–578.

[9] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[10] T. Zhu, M. A. Pimentel, G. D. Clifford, and D. A. Clifton, "Unsupervised bayesian inference to fuse biosignal sensory estimates for personalising care," *IEEE journal of biomedical and health informatics*, vol. 23, no. 1, p. 47, 2019.

[11] Y. Liu, W. Zhang, Y. Yu *et al.*, "Truth inference with a deep clustering-based aggregation model," *IEEE Access*, vol. 8, pp. 16 662–16 675, 2020.

[12] J. Huang, Y. Li, J. Tao, Z. Lian, M. Niu, and M. Yang, "Deep learning for continuous multiple time series annotations," in *Proceedings of the 2018 on Audio/Visual Emotion Challenge and Workshop*. ACM, 2018, pp. 91–98.

[13] Y. E. Kara, G. Genc, O. Aran, and L. Akarun, "Modeling annotator behaviors for crowd labeling," *Neurocomputing*, vol. 160, pp. 141–156, 2015.

[14] D. Tao, J. Cheng, Z. Yu, K. Yue, and L. Wang, "Domain-weighted majority voting for crowdsourcing," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 1, pp. 163–174, 2018.

[15] G. Rizos and B. W. Schuller, "Average jane, where art thou?–recent avenues in efficient machine learning under subjectivity uncertainty," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2020, pp. 42–55.

[16] P. Morales-Alvarez, P. Ruiz, S. Coughlin, R. M. Soriano, and A. Katsaggelos, "Scalable variational gaussian processes for crowdsourcing: Glitch detection in ligo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[17] J. Zhang, X. Wu, and V. S. Sheng, "Imbalanced multiple noisy labeling," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 489–503, 2014.

[18] A. P. Dawid and A. M. Skene, "Maximum likelihood estimation of observer error-rates using the em algorithm,"

Applied statistics, pp. 20–28, 1979.

[19] P. Ruiz, P. Morales-Álvarez, R. Molina, and A. K. Katsaggelos, "Learning from crowds with variational gaussian processes," *Pattern Recognition*, vol. 88, pp. 298–311, 2019.

[20] F. Rodrigues, F. C. Pereira, and B. Ribeiro, "Gaussian process classification and active learning with multiple annotators." in *ICML*, 2014, pp. 433–441.

[21] J. Gil, M. Álvarez, and Á. Orozco, "Automatic assessment of voice quality in the context of multiple annotations," in *EMBC*. IEEE, 2015, pp. 6236–6239.

[22] P. Groot, A. Birlutiu, and T. Heskes, "Learning from multiple annotators with Gaussian processes," in *ICANN*. Springer, 2011, pp. 159–164.

[23] F. Rodrigues, M. Lourenco, B. Ribeiro, and F. Pereira, "Learning supervised topic models for classification and regression from crowds," *IEEE transactions on PAMI*, 2017.

[24] F. Rodrigues, F. Pereira, and B. Ribeiro, "Sequence labeling with multiple annotators," *Machine learning*, vol. 95, no. 2, pp. 165–181, 2014.

[25] F. Rodrigues and F. C. Pereira, "Deep learning from crowds," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] M. Y. Guan, V. Gulshan, A. M. Dai, and G. E. Hinton, "Who said what: Modeling individual labelers improves classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] F. Rodrigues, F. Pereira, and B. Ribeiro, "Learning from multiple annotators: Distinguishing good from random labelers," *Pattern Recognition Letters*, vol. 34, no. 12, pp. 1428–1436, 2013.

[28] Y. Yan, R. Rosales, G. Fung, R. Subramanian, and J. Dy, "Learning from multiple annotators with varying expertise," *Machine learning*, vol. 95, no. 3, pp. 291–327, 2014.

[29] Y. Xie, J. Li, T. Zhu, and C. Liu, "Continuous-valued annotations aggregation for heart rate detection," *IEEE Access*, vol. 7, pp. 37 664–37 671, 2019.

[30] X. Wang and J. Bi, "Bi-convex optimization to learn classifiers from multiple biomedical annotations," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 14, no. 3, pp. 564–575, 2016.

[31] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi, "Community-based bayesian aggregation models for crowdsourcing," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 155–164.

[32] F. M. P. D. Rodrigues, "Probabilistic models for learning from crowdsourced data," Ph.D. dissertation, 2016.

[33] W. Tang, M. Yin, and C.-J. Ho, "Leveraging peer communication to enhance crowdsourcing," in *The World Wide Web Conference*. ACM, 2019, pp. 1794–1805.

[34] J. Surowiecki, *The wisdom of crowds*. Anchor, 2005.

[35] U. Hahn, M. von Sydow, and C. Merdes, "How communication can make voters choose less well," *Topics in cognitive science*, 2018.

[36] E. G. Rodrigo, J. A. Aledo, and J. A. Gámez, "Machine learning from crowds: A systematic review of its applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 2, p. e1288, 2019.

[37] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[38] S. Theodoridis, *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015.

[39] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information theory*, vol. 39, no. 3, pp. 930–945, 1993.

[40] M. A. Álvarez, L. Rosasco, N. D. Lawrence *et al.*, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[41] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian processes and kernel methods: A review on connections and equivalences," *arXiv preprint arXiv:1807.02582*, 2018.

[42] B. Schölkopf, A. J. Smola, F. Bach *et al.*, *Learning with kernels: support vector machines, regularization,*

optimization, and beyond. MIT press, 2002.

[43] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International conference on computational learning theory*. Springer, 2001, pp. 416–426.

[44] S. M. Kay and S. M. Kay, *Fundamentals of statistical signal processing: estimation theory*. Prentice-hall Englewood Cliffs, NJ, 1993, vol. 1.

[45] F. Rodrigues and F. C. Pereira, "Heteroscedastic gaussian processes for uncertainty modeling in large-scale crowdsourced traffic data," *Transportation research part C: emerging technologies*, vol. 95, pp. 636–651, 2018.

[46] A. N. Tikhonov and V. Y. Arsenin, "Solutions of ill-posed problems," *New York*, vol. 1, p. 30, 1977.

[47] T. Tian and J. Zhu, "Max-margin majority voting for learning from crowds," in *Advances in neural information processing systems*, 2015, pp. 1621–1629.

[48] D. E. Imbajoa-Ruiz, I. Gustin, M. Bolaños-Ledezma, A. F. Arciniegas-Mejía, F. Guasmayan-Guasmayan, M. Bravo-Montenegro, A. E. Castro-Ospina, and D. H. Peluffo-Ordóñez, "Multi-labeler classification using kernel representations and mixture of classifiers," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2016, pp. 343–351.

[49] P. Morales-Álvarez, P. Ruiz, R. Santos-Rodríguez, R. Molina, and A. K. Katsaggelos, "Scalable and efficient learning from crowds with Gaussian processes," *Information Fusion*, vol. 52, pp. 110–127, 2019.

[50] Y. Yan, R. Rosales, G. Fung, M. W. Schmidt, G. H. Valadez, L. Bogoni, L. Moy, and J. G. Dy, "Modeling annotator expertise: Learning when everybody knows a bit of something." in *AISTATS*, 2010, pp. 932–939.

[51] P. Zhang and Z. Obradovic, "Learning from inconsistent and unreliable annotators by a gaussian mixture model and bayesian information criterion," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 553–568.

[52] H. Xiao, H. Xiao, and C. Eckert, "Learning from multiple observers with unknown expertise," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 595–606.

[53] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *J. Mach. Learn. Res.*, vol. 13, no. Mar, pp. 795–828, 2012.

[54] J. Gil-Gonzalez, A. Alvarez-Meza, and A. Orozco-Gutierrez, "Learning from multiple annotators using kernel alignment," *Pattern Recognition Letters*, vol. 116, pp. 150–156, 2018.

[55] M. Gönen and E. Alpaydin, "Localized multiple kernel learning," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 352–359.

[56] A. Zhang and X. Gao, "Data-dependent kernel sparsity preserving projection and its application for semi-supervised classification," *Multimedia Tools and Applications*, pp. 1–17, 2018.

[57] T. Cohn and L. Specia, "Modelling annotator bias with multi-task gaussian processes: An application to machine translation quality estimation," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 32–42.

[58] J. Gil-Gonzalez, A. Orozco-Gutierrez, and A. Alvarez-Meza, "Learning from multiple inconsistent and dependent annotators to support classification tasks," *Neurocomputing*, vol. 423, pp. 236–247, 2021.

[59] A. Saul, J. Hensman, A. Vehtari, and N. Lawrence, "Chained Gaussian processes," in *Artificial Intelligence and Statistics*, 2016, pp. 1431–1440.

[60] J. Gil-González, A. Valencia-Duque, A. Álvarez-Meza, Á. Orozco-Gutiérrez, and A. García-Moreno, "Regularized chained deep neural network classifier for multiple annotators," *Applied Sciences*, vol. 11, no. 12, p. 5409, 2021.

[61] Y. Teh, M. Seeger, and M. Jordan, "Semiparametric latent factor models," in *AISTATS 2005-Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[62] J. Gil-Gonzalez, J.-J. Giraldo, A. Alvarez-Meza, A. Orozco-Gutierrez, and M. Alvarez, "Correlated chained gaussian processes for datasets with multiple annotators," *IEEE Transactions on Neural Networks and Learning Systems, early access*, 2021.

[63] J. C. Principe, *Information theoretic learning: Renyi's entropy and kernel perspectives*. Springer Science &

Business Media, 2010.

[64] M. Esmaeily, H. S. Yazdi, S. Abbassi, and R. Monsefi, "Hierarchical cooperation of experts in learning from crowds," in *ICCKE*. IEEE, 2016, pp. 211–217.

[65] G. Hua, C. Long, M. Yang, and Y. Gao, "Collaborative active visual recognition from crowds: A distributed ensemble approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 582–594, 2018.

[66] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *The Journal of Machine Learning Research*, vol. 12, 2011.

[67] M. Varma and B. R. Babu, "More generality in efficient multiple kernel learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 1065–1072.

[68] W. Liu, J. C. Principe, and S. Haykin, *Kernel adaptive filtering: a comprehensive introduction*. John Wiley & Sons, 2011, vol. 57.

[69] J. D. Arias-Londoño, J. I. Godino-Llorente, J. Gutiérrez-Arriola, V. Osma-Ruiz, and N. Sáenz-Lechón, "Automatic grbas assessment using complexity measures and a multiclass gmm-based detector," *Models and Analysis of Vocal Emissions for Biomedical Applications*, pp. 111–114, 2011.

[70] J. Zhang, X. Wu, and V. S. Sheng, "Imbalanced multiple noisy labeling," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 489–503, 2015.

[71] K. B. Petersen, M. S. Pedersen *et al.*, "The matrix cookbook," *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[72] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[73] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[74] J. A. Hernández-Muriel, J. B. Bermeo-Ulloa, M. Holguin-Londoño, A. M. Álvarez-Meza, and Á. A. Orozco-Gutiérrez, "Bearing health monitoring using relief-f-based feature relevance analysis and HMM," *Applied Sciences*, vol. 10, no. 15, p. 5170, 2020.

[75] M. Álvarez, D. Luengo, M. Titsias, and N. D. Lawrence, "Efficient multioutput Gaussian processes through variational inducing kernels," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 25–32.

[76] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.

[77] J. Hensman, A. G. Matthews, and Z. Ghahramani, "Scalable variational Gaussian process classification," *Proceedings of Machine Learning Research*, vol. 38, pp. 351–360, 2015.

[78] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.

[79] P. Moreno-Muñoz, A. Artés, and M. Alvarez, "Heterogeneous multi-output Gaussian process prediction," in *Advances in neural information processing systems*, 2018, pp. 6711–6720.

[80] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[81] L. v. d. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[82] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[83] M. A. Álvarez and N. D. Lawrence, "Computationally efficient convolved multiple output Gaussian processes," *The Journal of Machine Learning Research*, vol. 12, pp. 1459–1500, 2011.

[84] X. Liu, Y. Zou, Y. Song, C. Yang, J. You, and B. K Vijaya Kumar, "Ordinal regression with neuron stick-breaking for medical diagnosis," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[85] J. Han, P. Luo, and X. Wang, "Deep self-learning from noisy labels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5138–5147.

[86] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, "S4l: Self-supervised semi-supervised learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1476–1485.

[87] M. Artetxe, G. Labaka, and E. Agirre, "A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings," *arXiv preprint arXiv:1805.06297*, 2018.

[88] M. Lázaro-Gredilla and M. K. Titsias, "Variational heteroscedastic gaussian process regression," in *ICML*, 2011.

Appendices

Chapter A CCGPMA Supplementary Material

A.1 Derivation of CCGPMA Lower Bounds

$$\log p(\boldsymbol{Y}) = \log \int p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) p(\hat{\boldsymbol{f}}|\boldsymbol{u}) p(\boldsymbol{u}) d\hat{\boldsymbol{f}} d\boldsymbol{u},$$

$$= \log \int \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) q(\boldsymbol{u}) \frac{p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) p(\boldsymbol{u})}{\prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) q(\boldsymbol{u})} d\hat{\boldsymbol{f}} d\boldsymbol{u}. \tag{A.1}$$

Using Jensen's inequality, we obtain

$$\log p(\boldsymbol{Y}) \geq \int \left(\prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) q(\boldsymbol{u}) \right) \log \left(\frac{p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) \prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) p(\boldsymbol{u})}{\prod_{j=1}^{J} p(\boldsymbol{f}_j|\boldsymbol{u}) q(\boldsymbol{u})} \right) d\hat{\boldsymbol{f}} d\boldsymbol{u}, \tag{A.2}$$

$$\mathcal{L} = \prod_{j=1}^{J} \int q(\boldsymbol{f}_j) \log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) d\hat{\boldsymbol{f}} - \sum_{q=1}^{Q} \mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q)), \tag{A.3}$$

where $\mathbb{D}_{KL}(\cdot||\cdot)$ is the Kullback-Leibler–(KL) divergence and $q(\boldsymbol{f}_j)$ is defined as follows:

$$q(\boldsymbol{f}_j) = \int p(\boldsymbol{f}_j|\boldsymbol{u}) q(\boldsymbol{u}) d\boldsymbol{u} = \mathcal{N}(\boldsymbol{f}_j|\boldsymbol{\iota}_j, \boldsymbol{S}_j), \tag{A.4}$$

and

$$\boldsymbol{\iota}_j = \boldsymbol{K}_{\boldsymbol{f}_j \boldsymbol{u}} \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1} \boldsymbol{m}, \tag{A.5}$$

$$\boldsymbol{S}_j = \boldsymbol{K}_{\boldsymbol{f}_j \boldsymbol{f}_j} + \boldsymbol{K}_{\boldsymbol{f}_j \boldsymbol{u}} \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1} (\boldsymbol{V} - \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}) \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{u}}^{-1} \boldsymbol{K}_{\boldsymbol{u}\boldsymbol{f}_j}. \tag{A.6}$$

Moreover, note that the solution of the variational expectation (analytical or numerical) will depend on the form of the likelihood $p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$:

$$\mathbb{E}_{\prod_{j=1}^{J} q(f_j(\boldsymbol{x}_n))} \left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) \right] = \prod_{j=1}^{J} \int q(\boldsymbol{f}_j) \log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) d\hat{\boldsymbol{f}}, \tag{A.7}$$

For the numerical solution, we will use Gaussian-Hermite quadratures. Thus, considering for simplicity a univariate case, the expectations can be approximated as:

$$\mathbb{E}_{q(f_j)} \left[\log p(\boldsymbol{Y}|f_j) \right] \approx \frac{1}{\sqrt{\pi}} \sum_{s=1}^{S} o_s \log p(\boldsymbol{Y}|\sqrt{2\sigma_j} g_s + \iota_j), \tag{A.8}$$

where ι_j and σ_j are the mean and the variance of the distribution $q(f_j)$, respectively. Besides, the pair of values o_s and g_s are obtained from the Hermite polynomial $H_n(x) = (-1)^n e^{x^2} \frac{d}{dx^n} e^{-x^2}$.

A.1.1 Gradients w.r.t. the Variational Parameters

Here, we aim to maximize the lower bound in equation (A.3) for the variational parameters $\boldsymbol{m}_q$ and $\boldsymbol{V}_q$ of each distribution $q(\boldsymbol{u}_q)$.

Useful Equalities: These equalities are useful to compute the gradients [77]:

$$\frac{\partial}{\partial \iota}\mathbb{E}_{\mathcal{N}(x|\iota,\sigma^2)}\left[h(x)\right] = \mathbb{E}_{\mathcal{N}(x|\iota,\sigma^2)}\left[\frac{\partial}{\partial x}h(x)\right]. \tag{A.9}$$

$$\frac{\partial}{\partial \sigma^2}\mathbb{E}_{\mathcal{N}(x|\iota,\sigma^2)}\left[h(x)\right] = \frac{1}{2}\mathbb{E}_{\mathcal{N}(x|\iota,\sigma^2)}\left[\frac{\partial^2}{\partial x^2}h(x)\right]. \tag{A.10}$$

First, the bound derivatives w.r.t $\boldsymbol{m}_q$, yields:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{m}_q} = \underbrace{\frac{\partial}{\partial \boldsymbol{m}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]}_{\text{Variational Expectation–(VE) part}} \underbrace{-\frac{\partial}{\partial \boldsymbol{m}_q}\sum_{q=1}^{Q}\mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q))}_{\text{KL part}}, \tag{A.11}$$

where the KL part w.r.t. $\boldsymbol{m}_q$ is defined as:

$$\frac{\partial}{\partial \boldsymbol{m}_q}\mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q)) = \boldsymbol{K}_{\boldsymbol{u}_q\boldsymbol{u}_q}^{-1}\boldsymbol{m}_q. \tag{A.12}$$

Now, the VE part w.r.t. $\boldsymbol{m}_q$ is given as:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{m}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] &= \frac{\partial}{\partial \boldsymbol{m}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right], \\ &\quad \text{Chain Rule: } \boldsymbol{\iota}_j = h(\boldsymbol{m}_q), \\ &= \underbrace{\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial \hat{\boldsymbol{f}}}\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]}_{\text{See Likelihoods}}\frac{\partial \boldsymbol{\iota}_j}{\partial \boldsymbol{m}_q}.\end{aligned} \tag{A.13}$$

Second, the bound derivatives w.r.t $\boldsymbol{V}_q$ are defined as:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{V}_q} = \underbrace{\frac{\partial}{\partial \boldsymbol{V}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]}_{\text{VE part}} - \underbrace{\frac{\partial}{\partial \boldsymbol{V}_q}\sum_{q=1}^{Q}\mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q))}_{\text{KL part}}, \tag{A.14}$$

where the KL part w.r.t. $\boldsymbol{V}_q$, yields:

$$\frac{\partial}{\partial \boldsymbol{V}_q}\mathbb{D}_{KL}(q(\boldsymbol{u}_q)||p(\boldsymbol{u}_q)) = \frac{1}{2}\left(\boldsymbol{K}_{\boldsymbol{u}_q\boldsymbol{u}_q}^{-1} - \boldsymbol{V}_q^{-1}\right).$$

Now, the VE part w.r.t. $\boldsymbol{V}_q$ is given as:

$$\frac{\partial}{\partial \boldsymbol{V}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \frac{\partial}{\partial \boldsymbol{V}_q}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right],$$

$$\text{Chain Rule: } \boldsymbol{s}_j = h(\boldsymbol{V}_q),$$

$$= \frac{1}{2}\underbrace{\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]}_{\text{See Likelihoods}}\frac{\partial \boldsymbol{s}_j}{\partial \boldsymbol{V}_q}, \tag{A.15}$$

where $\boldsymbol{s}_j \in \mathbb{R}^N$ is a vector containing the diagonal elements of matrix $\boldsymbol{S}_j$. On the other hand, the gradients w.r.t. the hyper-parameters are similar to the ones derived in [79].

A.2 Likelihood Functions

The models exposed previously accept a wide variety of likelihood functions $p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$ [59]. To incorporate any new likelihood, we need to compute the following expressions:

1. The likelihood $p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$.
2. The Log-likelihood $\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$.
3. The variational expectation of the Log-likelihood $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$.
4. The first order derivatives $\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$.
5. The variational expectation of the first-order derivatives $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$.
6. The second order derivatives $\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$.
7. The variational expectation of the second-order derivatives $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$.
8. The predictive mean and variance.

A.2.1 Multiclass Classification with Multiple Annotators

To model categorical data from multiple annotators using our CCGPMA, we use the framework proposed in [27], which introduces a binary variable $\lambda_n^{(r)} \in \{0, 1\}$ representing the r-th labeler's reliability as a function of each sample $\boldsymbol{x}_n$. If $\lambda_n^{(r)} = 1$, the r-th annotator is supposed to provide the actual label, yielding a categorical distribution. Conversely, $\lambda_n^{(r)} = 0$ indicates that the r-th annotator gives an incorrect output, modeled by a uniform distribution.

1. $p(\boldsymbol{Y}|\boldsymbol{\theta})$ is written as:

$$p(\boldsymbol{Y}|\boldsymbol{\theta}) = \prod_{n=1}^{N}\prod_{r\in R_n}\left(\prod_{k=1}^{K}\zeta_{k,n}^{\delta(y_n^{(r)},k)}\right)^{\lambda_n^{(r)}}\left(\frac{1}{K}\right)^{(1-\lambda_n^{(r)})},$$

where $\delta(y_n^{(r)}, k) = 1$, if $y_n^{(r)} = k$, and $\delta(y_n^{(r)}, k) = 0$ otherwise. Moreover, $\zeta_{k,n} = p(y_n^{(r)} = k|\lambda_n^{(r)} = 1)$; note that since $\lambda_n^{(r)} = 1$, $\zeta_{k,n} = p(y_n = k)$ is an estimation of the unknown ground truth. Accordingly, to use our CCGPMA over this model, we need

$J = K + R$ LFs; K LFs to model $\zeta_{k,n}$, yielding:

$$\zeta_{k,n} = \Xi(f_k(\boldsymbol{x}_n)) = \frac{\exp(f_k(\boldsymbol{x}_n))}{\sum_{j=1}^{K}\exp(f_j(\boldsymbol{x}_n))}. \tag{A.16}$$

Besides, we need R LFs to perform a soft estimation of each $\lambda_n^{(r)}$; therefore, $\lambda_n^{(r)} = \varsigma(f_{l_r}(\boldsymbol{x}_n))$ with $r \in \{1, \dots R\}$, and $l_r = K + r \in \{K+1, \dots J\}$. Here, $\varsigma(f_{l_r}(\boldsymbol{x}_n))$ is the logistic sigmoid function: $\varsigma(a) = 1/(1+\exp(-a))$, where $a \in \mathbb{R}$.

2. $\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$

$$\begin{aligned}\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) &= \log \prod_{n=1}^{N}\prod_{r\in R_n}\left(\prod_{k=1}^{K}\zeta_{k,n}^{\delta(y_n^{(r)},k)}\right)^{\lambda_n^{(r)}}\left(\frac{1}{K}\right)^{(1-\lambda_n^{(r)})}, \\ &= \sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}.\end{aligned} \tag{A.17}$$

3. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$, yields:

$$\begin{aligned}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] &= \int\prod_{j=1}^{J} q(\hat{\boldsymbol{f}}_j)\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})d\hat{\boldsymbol{f}}, \\ &= \sum_{n=1}^{N}\sum_{r\in R_n}\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}]\sum_{k=1}^{K}\delta(y_n^{(r)},k)\mathbb{E}_{q(f_1,n)\dots q(f_K,n)}\left[\log\left(\zeta_{k,n}\right)\right] + \cdots \\ &\qquad \cdots + (1-\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}])\log\left(\frac{1}{K}\right),\end{aligned} \tag{A.18}$$

where

$$\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}] = \int q(f_{l_r,n})\varsigma(f_{l_r,n})df_{l_r,n}. \tag{A.19}$$

$$\mathbb{E}_{q(f_1,n)\dots q(f_K,n)}\left[\log\left(\zeta_{k,n}\right)\right] = \int q(f_{1,n})\dots q(f_{K,n})\log(\Xi\left(f_k(\boldsymbol{x}_n)\right))df_{1,n}\dots df_{K,n}. \tag{A.20}$$

The previous integrals do not have a close solution; hence, we approximate them by using the Gaussian-Hermite quadratures.

$$\mathbb{E}_{q(f_1,n)\dots q(f_K,n)}\left[\log\left(\zeta_{k,n}\right)\right] \approx \left(\frac{1}{\sqrt{\pi}}\right)^{K}\sum_{s_K=1}^{S_1}\cdots\sum_{s_1=1}^{S_K} o_{s_K}\dots o_{s_1}\log\left(\Xi\left(g_{s_k}\sqrt{2\sigma_{k,n}}+\iota_{k,n}\right)\right), \tag{A.21}$$

$$\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}] \approx \frac{1}{\sqrt{\pi}}\sum_{s=1}^{S} o_s\varsigma\left(g_s\sqrt{2\sigma_{l_r,n}}+\iota_{l_r,n}\right), \tag{A.22}$$

where

$$\Xi\left(g_{s_k}\sqrt{2\sigma_{k,n}}+\iota_{k,n}\right) = \frac{\exp(g_{s_k}\sqrt{2\sigma_{k,n}}+\iota_{k,n})}{\sum_{i=1}^{K}\exp(g_{s_i}\sqrt{2\sigma_{i,n}}+\iota_{i,n})}, \tag{A.23}$$

$$\varsigma\left(g_s\sqrt{2\sigma_{l_r,n}}+\iota_{l_r,n}\right) = \frac{1}{1+\exp(-(g_s\sqrt{2\sigma_{l_r,n}}+\iota_{l_r,n})}. \tag{A.24}$$

4. $\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$

$$\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}\right] \tag{A.25}$$

Case 1

$$\begin{aligned}\frac{\partial}{\partial f_{k,n}}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n})\right\}\right] &= \sum_{r\in R_n}\lambda_n^{(r)}\frac{\partial}{\partial f_{k,n}}\left[\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right)\right], \\ &= \sum_{r\in R_n}\lambda_n^{(r)}\left(\delta(y_n^{(r)},k)-\zeta_{k,n}\right). \end{aligned} \tag{A.26}$$

Case 2

$$\frac{\partial}{\partial f_{l_r,n}}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}\right] = \begin{cases}\Omega, & \text{if } r\in R_n \\ 0, & \text{Otherwise}\end{cases},$$

where

$$\begin{aligned}\Omega &= \frac{\partial\lambda_n^{(r)}}{\partial f_{l_r,n}}\left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n})\right) - \frac{\partial\lambda_n^{(r)}}{\partial f_{l_r,n}}\left(\log\left(\frac{1}{K}\right)\right), \\ &= \left(\lambda_n^{(r)}-(\lambda_n^{(r)})^2\right)\left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right)+\log(K)\right). \end{aligned} \tag{A.27}$$

5. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$

Case 1

$$\begin{aligned}\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{k,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] &= \sum_{r\in R_n}\mathbb{E}_{q(f_{1,n})\ldots q(f_{K,n})q(f_{l_r,n})}\left[\lambda_n^{(r)}\left(\delta(y_n^{(r)},k)-\zeta_{k,n}\right)\right], \\ &= \sum_{r\in R_n}\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]\left(\delta(y_n^{(r)},k)-\mathbb{E}_{q(f_{1,n})\ldots q(f_{K,n})}[\zeta_{k,n}]\right), \end{aligned} \tag{A.28}$$

where

$$\mathbb{E}_{q(f_1,n)\ldots q(f_K,n)}\left[\zeta_{k,n}\right] \approx \left(\frac{1}{\sqrt{\pi}}\right)^K \sum_{s_K=1}^{S_1}\cdots\sum_{s_1=1}^{S_K} o_{s_K}\ldots o_{s_1}\Xi\left(g_{s_k}\sqrt{2\sigma_{k,n}}+\iota_{k,n}\right), \tag{A.29}$$

and $\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]$ is approximated using equation (A.22),

Case 2

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{l_r,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \begin{cases}\mathbb{E}_{q(f_{1,n})\ldots q(f_{K,n})q(f_{l_r,n})}\left[\Omega\right], & \text{if } r\in R_n \\ 0, & \text{Otherwise}\end{cases},$$

where

$$\begin{aligned}
&\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\Omega\right] = \\
&= \mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\left(\lambda_n^{(r)} - (\lambda_n^{(r)})^2\right)\left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right) + \log(K)\right)\right], \\
&= \left(\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}] - \mathbb{E}_{q(f_{l_r,n})}[(\lambda_n^{(r)})^2]\right) \times \cdots \\
&\quad \cdots \times \left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\log\left(\zeta_{k,n}\right)\right] + \log(K)\right),
\end{aligned} \tag{A.30}$$

where $\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\log\left(\zeta_{k,n}\right)\right]$, and $\mathbb{E}_{q(f_{l_r,n})}[\lambda_n^{(r)}]$ are approximated by equations (A.21) and (A.22) respectively, and

$$\mathbb{E}_{q(f_{l_r,n})}[(\lambda_n^{(r)})^2] = \int q(f_{l_r,n})\left(\sigma(f_{l_r,n})\right)^2 df_{l_r,n} \approx \frac{1}{\sqrt{\pi}}\sum_{s=1}^{S} o_s\left[\varsigma\left(g_s\sqrt{2\sigma_{l_r,n}} + \iota_{l_r,n}\right)\right]^2. \tag{A.31}$$

6. $\dfrac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$

$$\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}\right]. \tag{A.32}$$

Case 1

$$\begin{aligned}
\frac{\partial^2}{\partial f_{k,n}^2}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}\right] &= \\
= \sum_{r\in R_n}\lambda_n^{(r)}\frac{\partial}{\partial f_{k,n}}\left[\delta(y_n^{(r)},k) - \zeta_{k,n}\right], &\\
= -\sum_{r\in R_n}\lambda_n^{(r)}(\zeta_{k,n} - \zeta_{k,n}^2) &
\end{aligned} \tag{A.33}$$

Case 2

$$\frac{\partial^2}{\partial f_{l_r,n}^2}\left[\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\lambda_n^{(r)}\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log(\zeta_{k,n}) + (1-\lambda_n^{(r)})\log\left(\frac{1}{K}\right)\right\}\right] = \begin{cases}\frac{\partial\Omega}{\partial f_{l_r,n}}, & \text{if } r\in R_n \\ 0, & \text{Otherwise}\end{cases},$$

where

$$\begin{aligned}
\frac{\partial\Omega}{\partial f_{l_r,n}} &= \frac{\partial}{\partial f_{l_r,n}}\left[\left(\lambda_n^{(r)} - (\lambda_n^{(r)})^2\right)\right]\left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right) + \log(K)\right), \\
&= \left(2\left(\lambda_n^{(r)}\right)^3 - 3\left(\lambda_n^{(r)}\right)^2 + \lambda_n^{(r)}\right)\left(\sum_{k=1}^{K}\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right) + \log(K)\right).
\end{aligned} \tag{A.34}$$

7. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\dfrac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$

Case 1

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial f_{k,n}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = -\sum_{r\in R_n}\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\lambda_n^{(r)}(\zeta_{k,n}-\zeta_{k,n}^2)\right],$$
$$= -\sum_{r\in R_n}\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\zeta_{k,n}\right]+\cdots$$
$$\cdots+\sum_{r\in R_n}\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\zeta_{k,n}^2\right], \tag{A.35}$$

where $\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]$, and $\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}[\zeta_{k,n}]$ are respectively approximated by equations (A.22) and (A.29), and

$$\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\zeta_{k,n}^2\right] \approx \left(\frac{1}{\sqrt{\pi}}\right)^K \sum_{s_K=1}^{S_1}\cdots\sum_{s_1=1}^{S_K} o_{s_K}\dots o_{s_1}\left[\Xi\left(g_{s_k}\sqrt{2\sigma_{k,n}}+\iota_{k,n}\right)\right]^2, \tag{A.36}$$

Case 2

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial f_{l_r,n}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \begin{cases}\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\frac{\partial\Omega}{\partial f_{l_r,n}}\right], & \text{if } r\in R_n\\ 0, & \text{Otherwise}\end{cases},$$

where

$$\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\frac{\partial\Omega}{\partial f_{l_r,n}}\right] =$$
$$=\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})q(f_{l_r,n})}\left[\left(2\left(\lambda_n^{(r)}\right)^3-3\left(\lambda_n^{(r)}\right)^2+\lambda_n^{(r)}\right)\left(\sum_{k=1}^K\delta(y_n^{(r)},k)\log\left(\zeta_{k,n}\right)+\log(K)\right)\right],$$
$$=\left(2\mathbb{E}_{q(f_{l_r,n})}\left[\left(\lambda_n^{(r)}\right)^3\right]-3\mathbb{E}_{q(f_{l_r,n})}\left[\left(\lambda_n^{(r)}\right)^2\right]+\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]\right)$$
$$\times\left(\sum_{k=1}^K\delta(y_n^{(r)},k)\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}\left[\log\left(\zeta_{k,n}\right)\right]+\log(K)\right), \tag{A.37}$$

where $\mathbb{E}_{q(f_{l_r,n})}\left[\lambda_n^{(r)}\right]$, $\mathbb{E}_{q(f_{l_r,n})}\left[\left(\lambda_n^{(r)}\right)^2\right]$, $\mathbb{E}_{q(f_{l_r,n})}\left[\left(\lambda_n^{(r)}\right)^3\right]$, and $\mathbb{E}_{q(f_{1,n})\dots q(f_{K,n})}[\log(\zeta_{k,n})]$ are approximated respectively by equations (A.22), (A.29) and (A.31); also,

$$\mathbb{E}_{q(f_{l_r,n})}[(\lambda_n^{(r)})^3] = \int q(f_{l_r,n})\left(\sigma(f_{l_r,n})\right)^3 df_{l_r,n} \approx \frac{1}{\sqrt{\pi}}\sum_{s=1}^S o_s\left[\varsigma\left(g_s\sqrt{2\sigma_{l_r,n}}+\iota_{l_r,n}\right)\right]^3. \tag{A.38}$$

8. The predictive distributions.

   Given a new sample $\boldsymbol{x}_*$, we are interested in the predictive distributions for the ground truth $\zeta_{k,*}$, and the labelers' reliabilities $\lambda_*^{(r)}$.

   Predictive distribution for $\zeta_{k,*}$ Each $\zeta_{k,n}$ is computed by applying the Softmax function over each function $f_{k,n}$, $\forall k \in \{1,\dots,K\}$. Accordingly, the mean for $\zeta_{k,*}$ can be

approximated as

$$\mathbb{E}[\zeta_{k*}] = \int \Xi(f_{k,*})q(f_{1,*})\dots q(f_{K,*})df_{1,*}\dots df_{K,*},$$
$$\approx \left(\frac{1}{\sqrt{\pi}}\right)^K \sum_{s_K=1}^{S_1}\cdots\sum_{s_1=1}^{S_K} o_{s_K}\dots o_{s_1}\Xi\left(g_{s_k}\sqrt{2\sigma_{k,*}}+\iota_{k,*}\right). \quad (A.39)$$

For the predictive variance of $\zeta_{k,*}$, we use the expression $\mathrm{Var}[\zeta_{k,*}] = \mathbb{E}[\zeta_{k,*}^2] - \mathbb{E}[\zeta_{k,*}]^2$; hence, we need to compute $\mathbb{E}[\zeta_{k,*}^2]$, which is given as

$$\mathbb{E}[\zeta_{k,*}^2] = \int \Xi(f_{k,*})^2 q(f_{1,*})\dots q(f_{K,*})df_{1,*}\dots df_{K,*},$$
$$\approx \left(\frac{1}{\sqrt{\pi}}\right)^K \sum_{s_K=1}^{S_1}\cdots\sum_{s_1=1}^{S_K} o_{s_K}\dots o_{s_1}\left[\Xi\left(g_{s_k}\sqrt{2\sigma_{k,*}}+\iota_{k,*}\right)\right]^2. \quad (A.40)$$

Predictive distribution for $\lambda_*^{(r)}$ In this case, $\boldsymbol{\lambda}_r$ is computed by applying a Sigmoid function over the latent functions $\varsigma(\boldsymbol{f}_{l_r}), \forall r$. Thus, the mean for $\varsigma(f_*)$ is given as

$$\mathbb{E}[\sigma(f_{l_r,*})] = \int \varsigma(f_{l_r,*})q(f_{l_r,*})df_{l_r,*} \approx \frac{1}{\sqrt{\pi}}\sum_{s=1}^{S} o_s\varsigma\left(g_s\sqrt{2\sigma_{l_r,*}}+\iota_{l_r,*}\right). \quad (A.41)$$

For the variance of $\varsigma(f_*)$, we use the expression $\mathrm{Var}[\varsigma(f_{l_r,*})] = \mathbb{E}[\varsigma(f_{l_r,*})^2] - \mathbb{E}[\varsigma(f_{l_r,*})]^2$; hence, we need to compute $\mathbb{E}[\varsigma(f_{l_r,*})^2]$, which is given as

$$\mathbb{E}[\varsigma(f_{l_r,*})^2] = \int \varsigma(f_{l_r,*})^2 q(f_{l_r,*})df_{l_r,*} \approx \frac{1}{\sqrt{\pi}}\sum_{s=1}^{S} o_s\left[\varsigma\left(g_s\sqrt{2\sigma_{l_r,*}}+\iota_{l_r,*}\right)\right]^2. \quad (A.42)$$

A.2.2 Gaussian Distribution for Regression with Multiple Annotators

For real-valued outputs, *e.g.*, $\mathcal{Y}\subset\mathbb{R}$, we follow the multi-annotator model used in [1, 22, 23, 52], where each output $y_n^{(r)}$ is considered to be a corrupted version of the hidden ground truth y_n.

1. $p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$

$$p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) = \prod_{n=1}^{N}\prod_{r\in R_n}\mathcal{N}(y_n^{(r)}|y_n, v_n^{(r)}), \quad (A.43)$$

where, $y_n = f_1(\boldsymbol{x}_n)$, and $v_n^{(r)} = \exp(f_{l_r}(\boldsymbol{x}_n))$, with $r\in\{1,\dots R\}$, and $l_r = r+1\in\{2,\dots J\}$; hence, the number of required LFs are $J = R+1$.

2. $\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})$

$$\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) = \log\prod_{n=1}^{N}\prod_{r\in R_n}\mathcal{N}(y_n^{(r)}|y_n, v_n^{(r)}),$$
$$= -\frac{1}{2}\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\log(2\pi)+\log(v_n^{(r)})+\frac{(y_n^{(r)}-y_n)^2}{v_n^{(r)}}\right\}, \quad (A.44)$$

where $\log v_n^{(r)} = \log(\exp(f_{l_r,n})) = f_{l_r,n}$

3. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \int \prod_{j=1}^{J} q(\hat{\boldsymbol{f}}_j) \log p(\boldsymbol{Y}|\hat{\boldsymbol{f}}) d\hat{\boldsymbol{f}},$$
$$= \sum_{n=1}^{N} \sum_{r \in R_n} \mathbb{E}_{q(f_{1,n})q(f_{l_r,n})}\left[\log \mathcal{N}(y_n^{(r)}|y_n, v_n^{(r)})\right]. \tag{A.45}$$

Since $q(f_{1,n})$ and $q(f_{l_r,1})$ obey Gaussian distributions, the above integrals can be solved analytically as in [88]. Thus, we have

$$\int q(f_{1,n})q(f_{l_r,n}) \log \mathcal{N}(y_n^{(r)}|y_n, v_n^{(r)}) df_{1,n} df_{l_r,n}$$
$$= \log \mathcal{N}\left(y_n^{(r)}|\iota_{1,n}, \exp\left(\iota_{l_r,n} - \frac{\sigma_{l_r,n}}{2}\right)\right) - \frac{\sigma_{l_r,n}}{4} - \frac{s_{1,n}\exp\left(-\iota_{l_r,n} + \frac{\sigma_{l_r,n}}{2}\right)}{2}. \tag{A.46}$$

Thus,

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \sum_{n=1}^{N} \sum_{r \in R_n} \log \mathcal{N}\left(y_n^{(r)}|\iota_{1,n}, \exp\left(\iota_{l_r,n} - \frac{\sigma_{l_r,n}}{2}\right)\right) - \cdots$$
$$\cdots - \frac{\sigma_{l_r,n}}{4} - \frac{s_{1,n}\exp\left(-\iota_{l_r,n} + \frac{\sigma_{l_r,n}}{2}\right)}{2}, \tag{A.47}$$

where $f_{1,n}$ is the n-th position of vector $\boldsymbol{f}_1$, $f_{l_r,n}$ is the n-th position of vector $\boldsymbol{f}_{l_r}$, $\iota_{1,n}$ is the n-th position of vector $\boldsymbol{\iota}_1$, $\iota_{l_r,n}$ is the n-th position of vector $\boldsymbol{\iota}_{l_r}$, $s_{1,n}$ is the n-th position of vector $\boldsymbol{s}_1$, and $\sigma_{l_r,n}$ is the n-th position of vector $\boldsymbol{s}_{l_r}$.

4. $\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$

$$\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r \in R_n}\left\{\log(2\pi) + f_{l_r,n} + \frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right]. \tag{A.48}$$

Case 1

$$\frac{\partial}{\partial f_{1,n}}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r \in R_n}\left\{\frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right] = \sum_{r \in R_n}\frac{(y_n^{(r)} - f_{1,n})}{\exp(f_{l_r,n})}. \tag{A.49}$$

Case 2

$$\frac{\partial}{\partial f_{l_r,n}}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r \in R_n}\left\{f_{l_r,n} + \frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right] = \begin{cases} -\frac{1}{2} + \frac{(y_n^{(r)} - f_{1,n})^2}{2\exp(f_{l_r,n})}, & \text{if } r \in R_n \\ 0, & \text{Otherwise} \end{cases}. \tag{A.50}$$

5. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial \hat{\boldsymbol{f}}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$

Case 1

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{1,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \sum_{r \in R_n}\mathbb{E}_{q(f_{1,n})q(f_{l_r,n})}\left[\frac{(y_n^{(r)} - f_{1,n})}{\exp(f_{l_r,n})}\right],$$
$$= \sum_{r \in R_n}\left(y_n^{(r)} - \mathbb{E}_{q(f_{1,n})}[f_{1,n}]\right)\mathbb{E}_{q(f_{l_r,n})}\left[\exp(-f_{l_r,n})\right], \tag{A.51}$$

where

$$\mathbb{E}_{q(f_{1,n})}[f_{1,n}] = \iota_{1,n}, \tag{A.52}$$

$$\mathbb{E}_{q(f_{l_r,n})}\left[\exp(-f_{l_r,n})\right] = \exp\left(-\iota_{l_r,n} + \frac{\sigma_{l_r,n}}{2}\right). \tag{A.53}$$

Thus,

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{1,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \sum_{r\in R_n}\left(y_n^{(r)} - \iota_{1,n}\right)\exp\left(-\iota_{l_r,n} + \frac{\sigma_{l_r,n}}{2}\right). \tag{A.54}$$

Case 2

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{l_r,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \\ = \begin{cases} \frac{1}{2}\mathbb{E}_{q(f_{1,n})}\left[(y_n^{(r)} - f_{1,n})^2\right]\mathbb{E}_{q(f_{l_r,n})}\left[\exp(-f_{l_r,n})\right] - \frac{1}{2}, & \text{if } r \in R_n \\ 0, & \text{Otherwise} \end{cases}, \tag{A.55}$$

where

$$\mathbb{E}_{q(f_{1,n})}\left[(y_n^{(r)} - f_{1,n})^2\right] = \mathbb{E}_{q(f_{1,n})}\left[(y_n^{(r)})^2 - 2y_n^{(r)}f_{1,n} - (f_{1,n})^2\right], \\ = \left(y_n^{(r)} - \iota_{1,n}\right)^2 + s_{1,n}. \tag{A.56}$$

Furthermore, $\mathbb{E}_{q(f_{l_r,n})}\left[\exp(-f_{l_r,n})\right]$ is computed as in equation (A.53). Thus,

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial}{\partial f_{l_r,n}}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \\ = \begin{cases} \frac{1}{2}\left(\left(y_n^{(r)} - \iota_{1,n}\right)^2 + s_{1,n}\right)\exp\left(-\iota_{l_r,n} + \frac{\sigma_{l_r,n}}{2}\right) - \frac{1}{2}, & \text{if } r \in R_n \\ 0, & \text{Otherwise} \end{cases}. \tag{A.57}$$

6. $\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]$

$$\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right] = \frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\log(2\pi) + f_{l_r,n} + \frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right]. \tag{A.58}$$

Case 1

$$\frac{\partial^2}{\partial f_{1,n}^2}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\log(2\pi) + f_{l_r,n} + \frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right] = -\sum_{r\in R_n}\exp(-f_{l_r,n}). \tag{A.59}$$

Case 2

$$\frac{\partial^2}{\partial f_{l_r,n}^2}\left[-\frac{1}{2}\sum_{n=1}^{N}\sum_{r\in R_n}\left\{\log(2\pi) + f_{l_r,n} + \frac{(y_n^{(r)} - f_{1,n})^2}{\exp(f_{l_r,n})}\right\}\right] = \\ \begin{cases} \frac{-(y_n^{(r)} - f_{1,n})^2}{2\exp(f_{l_r,n})}, & \text{if } r \in R_n \\ 0, & \text{Otherwise} \end{cases}. \tag{A.60}$$

7. $\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial \hat{\boldsymbol{f}}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right]$

Case 1

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial f_{1,n}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \sum_{r\in R_n}\mathbb{E}_{q(f_{l_r,n})}\left[-\exp(-f_{l_r,n})\right], \\ = -\sum_{r\in R_n}\exp\left(-\iota_{l_r,n}+\frac{\sigma_{l_r,n}}{2}\right). \tag{A.61}$$

Case 2

$$\mathbb{E}_{q(\hat{\boldsymbol{f}})}\left[\frac{\partial^2}{\partial f_{l_r,n}^2}\left[\log p(\boldsymbol{Y}|\hat{\boldsymbol{f}})\right]\right] = \\ = \begin{cases} -\frac{1}{2}\mathbb{E}_{q(f_{1,n})}\left[(y_n^{(r)}-f_{1,n})^2\right]\mathbb{E}_{q(f_{l_r,n})}\left[\exp(-f_{l_r,n})\right], & \text{if } r\in R_n \\ 0, & \text{Otherwise} \end{cases}, \\ = \begin{cases} -\frac{1}{2}\left(\left(y_n^{(r)}-\iota_{1,n}\right)^2+s_{1,n}\right)\exp\left(-\iota_{l_r,n}+\frac{\sigma_{l_r,n}}{2}\right), & \text{if } r\in R_n \\ 0, & \text{Otherwise} \end{cases}. \tag{A.62}$$

8. The predictive distributions.

   Given a new sample $\boldsymbol{x}_*$, we are interested in the predictive distributions for the ground truth y_*, and the labelers' error-variances $v_*^{(r)}$.

   Predictive distribution for y_* Due to $\mathbf{y} = \mathbf{f}_1$, the posterior distribution for y_* corresponds to $q(f_{1*})$. Accordingly,

$$\mathbb{E}[y_*] = \mathbb{E}[f_{1,*}] = \iota_{1,*} \tag{A.63}$$

$$\mathrm{Var}[y_*] = \mathrm{Var}[f_{1,*}] = s_{1,*} \tag{A.64}$$

   Predictive distribution for $v_*^{(r)}$ In this case, since $\mathbf{v}_r = \exp(\mathbf{f}_{l_r})$, the posterior distribution for $v_*^{(r)}$ follows a log-normal distribution with parameters $\iota_{l_r,*}$ and $\sigma_{l_r,*}$, which respectively correspond to the mean and variance of $q(f_{l_r,*})$. In this sense, the mean and variance of $v_*^{(r)}$ are given as

$$\mathbb{E}[v_*^{(r)}] = \int \exp(f_{l_r,*})q(f_{l_r,*})df_{l_r,*} = \exp\left(\iota_{l_r,*}+\frac{\sigma_{l_r,*}}{2}\right). \tag{A.65}$$

$$\mathrm{Var}[v_*^{(r)}] = \int\left(\exp(f_{l_r,*})-\mathbb{E}[v_*^{(r)}]\right)^2 q(f_{l_r,*})df_{l_r,*} \\ = \exp\left(2\iota_{l_r,*}+\sigma_{l_r,*}\right)\left(\exp(\sigma_{l_r,*})-1\right). \tag{A.66}$$

Alphabetical Index

A

annotator relevance analysis 17
annotator's behavior 4
annotator's dependencies 5
annotators 4
annotators' relevance parameters 19

B

binary classification 4

C

Chained Gaussian Processes for Multiple Annotators (CGPMA) 21
classification problems 13
classification results 35
convex combination 39
covariance matrix 15
crowdsourcing 2

D

dataset 48
dataset results 48
dependencies 48
deterministic function 20, 60

E

error variances 88

F

fully real dataset 30
fully synthetic dataset 31
function 3
function g_r 26

G

Gaussian distribution 7, 73
Gaussian process 34, 47
gold standard 2, 10
ground truth 10, 31, 40

I

input 3, 11
input features 11
input samples 13
input space 11

T

target classifier 35, 49

U

upper bound 34

The increasing popularity of crowdsourcing platforms, i.e., Amazon Mechanical Turk, is changing how datasets for supervised learning are built. In these cases, instead of having datasets labeled by one source (which is supposed to be an expert who provided the absolute gold standard), we have datasets labeled by multiple annotators with different and unknown expertise. Hence, we face a multi-labeler scenario, which typical supervised learning models cannot tackle. For this reason, much attention has recently been given to the approaches that capture multiple annotators' wisdom. However, such methods reside on two key assumptions: the labeler's performance does not depend on the input space and independence among the annotators, which are hardly feasible in real-world settings. This book explores several models based on both frequentist and Bayesian perspectives aiming to face multi-labeler scenarios. Our approaches model the annotators' behavior by considering the relationship between the input space and the labelers' performance and coding interdependencies among them.