Multi-class Gaussian Process Classification with Noisy Inputs

Carlos Villacampa-Calvo   CARLOS.VILLACAMPA@UAM.ES
Computer Science Department, Universidad Autónoma de Madrid, 28049, Madrid, Spain
Bryan Zaldívar   BRYAN.ZALDIVARM@UAM.ES
Theoretical Physics Department, Universidad Autónoma de Madrid, 28049, Madrid, Spain
Instituto de Física Teórica, 28049, Madrid, Spain
Eduardo C. Garrido-Merchán   EDUARDO.GARRIDO@UAM.ES
Computer Science Department, Universidad Autónoma de Madrid, 28049, Madrid, Spain
Daniel Hernández-Lobato   DANIEL.HERNANDEZ@UAM.ES
Computer Science Department, Universidad Autónoma de Madrid, 28049, Madrid, Spain

Editor: Philipp Hennig

Abstract

It is a common practice in the machine learning community to assume that the observed data are noise-free in the input attributes. Nevertheless, scenarios with input noise are common in real problems, as measurements are never perfectly accurate. If this input noise is not taken into account, a supervised machine learning method is expected to perform sub-optimally. In this paper, we focus on multi-class classification problems and use Gaussian processes (GPs) as the underlying classifier. Motivated by a data set coming from the astrophysics domain, we hypothesize that the observed data may contain noise in the inputs. Therefore, we devise several multi-class GP classifiers that can account for input noise. Such classifiers can be efficiently trained using variational inference to approximate the posterior distribution of the latent variables of the model. Moreover, in some situations, the amount of noise can be known before-hand. If this is the case, it can be readily introduced in the proposed methods. This prior information is expected to lead to better performance results. We have evaluated the proposed methods by carrying out several experiments, involving synthetic and real data. These include several data sets from the UCI repository, the MNIST data set and a data set coming from astrophysics. The results obtained show that, although the classification error is similar across methods, the predictive distribution of the proposed methods is better, in terms of the test log-likelihood, than the predictive distribution of a classifier based on GPs that ignores input noise.

Keywords: Gaussian processes, Multi-class classification, Input dependent noise

1. Introduction

Multi-class classification problems involve predicting a class label y that can take values in a discrete set of C labels $\{1, \ldots, C\}$ with $C > 2$ (Murphy, 2012). For this task, one should use the information contained in the input attributes $\mathbf{x} \in \mathbb{R}^d$, with d the dimensionality of the data. That is, we assume input attributes in the real line. In order to infer the relation between y and $\mathbf{x}$ it is assumed that one can use some training data in the form of N pairs

$(\mathbf{x}, y)$, namely, $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. Multi-class classification problems arise in a huge variety of fields, from industry to science. Most of the times, however, it is common to have data sets whose inputs $\mathbf{x}$ are the result of experimental measurements. These measurements are unavoidably contaminated with noise, as a consequence of measurement error. Furthermore, in some situations, the errors in the measurement of the explaining attributes can be well determined. See Barford (1985) for further details. Incorporating this inductive bias, or prior knowledge about the particular characteristics of the inputs of the classification problem is expected to lead to better results when training a classifier to predict y given $\mathbf{x}$. Conversely, ignoring errors or noise in the input measurements is expected to lead to sub-optimal results when this is indeed the case. In this research work we have empirically validated this hypothesis on several commonly used data sets extracted from the UCI repository (Bay et al., 2000), as well as on the well-known MNIST data set (LeCun et al., 1998). Even for these commonly used data sets in the machine learning community, we find better prediction results by assuming the presence (and learning the amount) of noise in some of the corresponding input variables, as we will show later.

While inference tasks on data with noisy attributes have been considered since long time in the context of regression—see for example W.H. et al. (2007), or more recently Mchutchon and Rasmussen (2011), in the context of Gaussian processes—the specific case of multi-class classification has received much less attention from the literature, with a few exceptions (Sáez et al., 2014). Taking into account the presence of noise in the input is, as we show below, potentially essential to better modeling the conditional distribution $p(y|\mathbf{x})$ giving rise to the observed labels. Considering input noise is also expected to have an important impact on the classification of points that are not far from the decision boundaries, since those are regions of the input space in which the data is more susceptible of being misclassified (at least in the case of additive noise with finite variance). Needless to say, both improving the estimated underlying predictive distribution and the better confidence in the classification of *difficult* points are two desirable properties for real-world problems in generic science applications, as for example in the case of the medical domain or in astrophysics applications (Gal, 2016).

Gaussian Processes (GPs) are machine learning methods that are inherently specified within a Bayesian framework (Rasmussen and Williams, 2006). Therefore, they can deliver probabilistic outputs that allow to extract uncertainty in the predictions made. This uncertainty is specified in terms of a predictive distribution for the target variable and may arise both from the modeling of intrinsic noise and also due to the lack of observed data. GPs are also non-parametric models. Therefore, their expressiveness grows as the number of data points in the training set, of size N, increases. GPs are, however, expensive to train since their complexity is in $\mathcal{O}(N^3)$. Specifically, they require the inversion of a covariance matrix of size $N \times N$. These methods also suffer from the difficulty of requiring approximate techniques to compute the posterior distribution of the latent variables of the model in the case of classification tasks. This posterior distribution is required to compute the predictive distribution for the target variable. Nevertheless, in spite of this difficulties, GPs have been successfully used to address multi-class classification problems and have been shown to be competitive with other approaches such as support vector machines or neural networks (Kim and Ghahramani, 2006; Hernández-Lobato et al., 2011; Henao and Winther, 2012; Hensman et al., 2015b; Villacampa-Calvo and Hernández-Lobato, 2017).

To alleviate the problems of scalability of GPs several approximations have been proposed in the literature. Among them, the most popular ones are based on using inducing points representations (Snelson and Ghahramani, 2006; Titsias, 2009). These techniques consist in introducing a set of $M \ll N$ inducing points that are carefully chosen to approximate the full GP model. Specifically, the locations of the inducing points are optimized during training alongside with any other GP hyper-parameter, by maximizing an estimate of the marginal likelihood (Hensman et al., 2015b; Villacampa-Calvo and Hernández-Lobato, 2017). The complexity of these approximations is in $\mathcal{O}(NM^2)$, which is significantly better than $\mathcal{O}(N^3)$ when $M \ll N$. Importantly, sparse approximations based on inducing points can be combined with stochastic optimization techniques, which allow to scale GPs to very large data sets (Hensman et al., 2015a; Villacampa-Calvo and Hernández-Lobato, 2017). Nevertheless, in spite of this, to our knowledge, all current methods for multi-class GPs classification assume noiseless input attributes. In this paper we extend the framework of multi-class GP classification to account for this type of noise.

Our work is motivated by a concrete multi-class classification problem and from a data set coming from astrophysics, dealing with measurements, by the Fermi-LAT instrument operated by NASA, of point-like sources of photons in the gamma ray energy range all over the sky (`https://fermi.gsfc.nasa.gov/`). The experimental measurements obtained from such a system are unavoidably contaminated with noise and, in practice, the level of noise in some of the explaining attributes is known and can be well determined. As it turns out, at present, a significant fraction of those point-like sources are not associated to or labeled as known astrophysical gamma rays sources as for example pulsars, blazars or quasars (Abdollahi et al., 2020). It is thus of paramount importance for the physics related to those measurements to know whether those point-like sources belong to the standard astrophysical classes, or instead whether they are part of other more exotic kind of sources. As a study-case exercise, though, we train a fully supervised classifier using exclusively the sources which do have labels already. Further details of the data set are given when presenting our experimental results. These show that a GP multi-class classifier which considers input noise can obtain better predictive distributions in terms of the test log-likelihood in this data set.

In this paper we focus on multi-class classification problems. The reason for this is partially that this is precisely the setting of the data set coming from astrophysics that motivated this work. Besides this, multi-class classification using GPs is relevant since it has systematically received less attention from the machine learning community than other settings such as regression or binary classification (Kuss and Rasmussen, 2005; Titsias, 2009; Hensman et al., 2013). This is probably related to the fact that addressing multi-class problems with GPs is more challenging. More precisely, in a multi-class setting there is one latent function per class label that has to be modeled. By contrast, in the regression and binary classification settings in there is only one latent function. Multi-class problems also involve the difficulty of dealing with complicated likelihood factors which often have the form of soft-max functions or Gaussian integrals that need to be approximated and that may even lack a closed form expression. See, *e.g.*, Villacampa-Calvo and Hernández-Lobato (2017). In spite of this, we also give some intuitions about how to extend the proposed approach to other supervised learning settings different from multi-class classification.

To account for input noise in the context of multi-class GP classification, we describe three different methods. A first approach is based on considering the actual noiseless input attributes (which we denote by $\mathbf{x}$) as a latent variable, to then perform approximate Bayesian inference and compute their posterior distribution. The required computations are approximated using variational inference (Blei et al., 2017) combined with Monte Carlo methods and stochastic optimization. A second approach considers a first order Taylor expansion of the predictive mean of the GPs contained in the multi-class GP classifier (typically, one different GP per each potential class label), following Mchutchon and Rasmussen (2011). Under this linear approximation the input noise is simply translated into output noise, which is incorporated in the modeling process. The variance of the output noise is determined by the slope of the GP predictive mean, which can be obtained using automatic differentiation tools. Variational inference is also used in this second method to approximate the required computations. The two methods described to account for input noise in the context of multi-class GP classification are validated on several experiments, including synthetic data, data sets from the UCI repository, MNIST, and the data set related to astrophysics that motivated this work described above. These experiments give empirical claims supporting our hypothesis that our methods can effectively deal with noise in the inputs. In particular, we have consistently observed that the predictive distribution of the methods proposed is better, in terms of the test log-likelihood, than the one of a standard multi-class GP classifier that ignores input noise. The prediction error of the proposed method is, however, similar. A better predictive distribution means that in general the uncertainty about the potential class label of a data instance is better modeled. We illustrate that this is indeed the case in a set of active learning experiments. In these experiments one chooses the most informative points from a validation set to be labeled and included in the training set with the goal of reducing the prediction error. The most informative points are those for which the model prediction is most uncertain. Our results confirm that by taking into account input noise a better prediction error is obtained in these experiments.

The rest of the manuscript is organized as follows: We first introduce the fundamentals about multi-class GP classification and sparse GPs in Section 2. Section 3 describes the proposed models and methods to account for input noise in the observed attributes. Related work about input noise in GPs and other machine learning methods is described in Section 4. Section 5 illustrates the empirical performance of the proposed methods. Finally, Section 6 gives the conclusions of this work.

2. Multi-class Gaussian Process Classification

In this section we describe how Gaussian processes (GPs) can be used to address multi-class classification problems. We consider first a noiseless input setting. Next, in the following section, we describe how noisy inputs can be incorporated into the model. Assume a data set consisting of N instances with $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^{\mathrm{T}}$ the observed explaining attributes and $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ the target class labels, where $y_i \in \{1, \ldots C\}$ and $C > 2$ is the number of classes. The task of interest is to make predictions about the label y^* of a new instance $\mathbf{x}^{\star}$ given the observed data $\mathbf{X}$ and $\mathbf{y}$.

Following the representation introduced by Kim and Ghahramani (2006) for multi-class classification with GPs, we assume that each class label has been obtained with the labeling

rule

$$y_i = \arg \max_c \quad f^c(\mathbf{x}_i) \,, \tag{1}$$

where $f^c(\cdot)$, for $c = 1, \ldots, C$, are different latent functions, each one of them corresponding to a different class label. Therefore the class label has been obtained simply by considering the latent function with the largest value at the data point $\mathbf{x}_i$. Let $\mathbf{f}_i = (f^1(\mathbf{x}_i), \ldots, f^C(\mathbf{x}_i))^{\mathrm{T}}$. Under this labeling rule the likelihood of the value of each latent function at a training point is given by

$$p(y_i | \mathbf{f}_i) = \prod_{c \neq y_i} \Theta \left(f^{y_i}(\mathbf{x}_i) - f^c(\mathbf{x}_i) \right) \,,$$

where $\Theta(\cdot)$ is a Heaviside step function. Other likelihood functions such as the soft-max likelihood arise simply by considering and marginalizing Gumbel noise around the latent functions $f^c(\cdot)$ (Maddison et al., 2014). Here we instead consider Gaussian noise around each f^c, as described later on. To account for labeling errors, we consider that the actual class label y_i associated to $\mathbf{x}_i$ could have been flipped with probability ϵ to some other class label (Hernández-Lobato et al., 2011; Hensman et al., 2015b). Under this setting, the likelihood becomes

$$p(y_i | \mathbf{f}_i) = (1 - \epsilon) \prod_{c \neq y_i} \Theta \left(f^{y_i}(\mathbf{x}_i) - f^c(\mathbf{x}_i) \right) + \frac{\epsilon}{C - 1} \left[1 - \prod_{c \neq y_i} \Theta \left(f^{y_i}(\mathbf{x}_i) - f^c(\mathbf{x}_i) \right) \right] \,. \tag{2}$$

In order to address multi-class classification with GPs, a GP prior is assumed for each latent function $f^c(\cdot)$ (Rasmussen and Williams, 2006). That is, $p(f^c) \sim \mathcal{GP}(0, k_\theta(\cdot, \cdot))$, where $k_{\theta_c}(\cdot, \cdot)$ is a covariance function, with hyper-parameters θ_c. Popular examples of covariance functions include the squared exponential covariance function. Namely,

$$k_{\theta_c}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp \left\{ -\frac{1}{2} \sum_{j=1}^d \frac{(x_j - x_j')^2}{\ell_j} \right\} + I[\mathbf{x} = \mathbf{x}'] \sigma_0^2 \,,$$

where $I[\cdot]$ is an indicator function and $\theta_c = \{\sigma^2, \sigma_0, \{\ell_j\}_{j=1}^d\}$ are the hyper-parameters. More precisely, σ^2 is the amplitude parameter, ℓ_j are the length-scales and σ_0^2 is the level of additive Gaussian noise around f^c. See Rasmussen and Williams (2006) for further details. In practice, the hyper-parameters will be different for each latent function $f^c(\cdot)$. We have ignored here their dependence on the latent function $f^c(\cdot)$ for the sake of readability.

In order to make predictions about the potential class label of a new data point $\mathbf{x}^\star$, one would like to compute the posterior distribution of $\mathbf{f} = \{\mathbf{f}_i\}_{i=1}^N$. This distribution summarizes which values of the latent functions are compatible with the observed data. The posterior distribution can be computed using Bayes' rule as follows:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y})} = \frac{\left[\prod_{i=1}^N p(y_i|\mathbf{f}_i) \right] \left[\prod_{c=1}^C p(\mathbf{f}^c) \right]}{p(\mathbf{y})} \,. \tag{3}$$

where $\mathbf{f}^c = (f_c(\mathbf{x}_1), \ldots, f_c(\mathbf{x}_N))^{\mathrm{T}}$ and $p(\mathbf{f}^c) = \mathcal{N}(\mathbf{f}^c|\mathbf{0}, \mathbf{K}^c)$ is a multi-variate Gaussian distribution with zero mean and covariance matrix $\mathbf{K}^c$, with $K_{i,j}^c = k_{\theta_c}(\mathbf{x}_i, \mathbf{x}_j)$. The denominator in the previous expression, $p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})d\mathbf{f}$, is just a normalization constant and is known as the marginal likelihood. It can be maximized to obtain good values for the model hyper-parameters θ_c, for $c = 1, \ldots, C$. Note that in this setting, we assume independence among the latent functions $f^c(\cdot)$, since the prior factorizes as $p(\mathbf{f}) = \prod_{c=1}^{C} p(\mathbf{f}^c)$.

A practical problem, however, is that the non-Gaussian likelihood in (2) makes infeasible the exact computation of the marginal likelihood, so one has to make use of approximate inference methods to approximate the posterior in (3). One of the most widely used methods for approximate inference is variational inference (VI), which will be explained in detail in the following sections (Titsias, 2009; Hensman et al., 2015a,b). In VI (3) is approximated by a Gaussian distribution q whose parameters are obtained by minimizing the Kullback-Leibler (KL) divergence between q and the exact posterior. The main advantage of VI is that it transforms the approximate inference problem into an optimization problem that can be solved using stochastic optimization techniques, and can hence scale to large data sets. Furthermore, it can be easily implemented in modern frameworks for machine learning such as Tensorflow (Abadi et al., 2015).

Other techniques that have been used for approximate inference in the context of multi-class GP classification include the Laplace approximation (Williams and Barber, 1998). This technique finds a Gaussian approximation matching the curvature of the posterior at its mode. Nevertheless, it scales poorly to large data sets, since finding the model hyper-parameters by approximately maximizing the marginal likelihood requires a double loop algorithm (Rasmussen and Williams, 2006), and there is evidence showing that other methods (such as expectation propagation) give better results, at least for binary classification (Nickisch and Rasmussen, 2008). Expectation propagation (EP) has also been used for approximate inference in multi-class GP classification (Riihimäki et al., 2013; Villacampa-Calvo and Hernández-Lobato, 2017). This method is competitive with VI and sometimes gives better predictive distributions. Unlike VI, EP approximately minimizes the KL-divergence between the posterior (3) and an approximate Gaussian distribution q. The disadvantage is that it is based on a set of update rules to find q and not on solving directly an optimization problem. These update rules are difficult to implement in frameworks such as Tensorflow.

The KL-divergence is generalized by a set of divergences known as the α-divergence (Bui et al., 2017). The use of these divergences for approximate inference in multi-class GP classification has been investigated by Villacampa-Calvo and Hernández-Lobato (2020), showing that sometimes one can obtain better results than those of VI or EP. In this work we focus on VI for approximate inference because of its simplicity and ease of implementation. The potential use of α-divergences for approximate inference is left as future work.

2.1 Sparse Gaussian Processes

A difficulty of the method described so far is that, even if the likelihood was Gaussian and exact inference was tractable, the cost of computing the posterior distribution would be in $\mathcal{O}(N^3)$, where N is the training set size. The reason for this cost is the need of inverting the prior covariance matrices $\mathbf{K}^c$. See Rasmussen and Williams (2006) for further details.

As a consequence, multi-class GPs classification would only be applicable on a data set of at most a few thousand data points.

A popular and successful approach to reduce the previous cost, is to consider an approximation based on sparse Gaussian processes (Titsias, 2009). Under a sparse setting, a set of M pseudo-inputs or inducing points is introduced associated to each latent function $f^c(\cdot)$. Namely, $\mathbf{Z}^c = (\mathbf{z}_1^c, \ldots, \mathbf{z}_M^c)$. These points $\mathbf{Z}^c$ will lie in the same space as the training data. Namely, $\mathbb{R}^d$, and their locations will be specified during training, simply by maximizing an estimate of the marginal likelihood. Associated to these inducing points $\mathbf{Z}^c$ we will consider some inducing outputs $\mathbf{u}^c$, where $u_j^c = f^c(\mathbf{z}_j^c)$. The process values at each $\mathbf{x}_i$, *i.e.*, $\mathbf{f}^c = (f^c(\mathbf{x}_1), \ldots, f^c(\mathbf{x}_N))^{\mathrm{T}}$, are characterized by $\mathbf{Z}^c$ and $\mathbf{u}^c$ and can then be obtained from the predictive distribution of a GP as follows:

$$p(\mathbf{f}^c|\mathbf{u}^c) = \mathcal{N}\left(\mathbf{f}^c|\mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{u}^c, \mathbf{K}_{\mathbf{X},\mathbf{X}}^c - \mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{K}_{\mathbf{Z}^c,\mathbf{X}}^c\right), \qquad (4)$$

where $\mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c$ is a $N \times M$ matrix of covariances of $f^c(\cdot)$ between the process values at the observed data points $\mathbf{X}$ and the inducing points $\mathbf{Z}^c$. Similarly, $\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c$ is the $M \times M$ covariance matrix. Each entry in this matrix contains the covariances among the process values at the inducing points $\mathbf{Z}^c$. Under the sparse approximation, the prior for each $\mathbf{u}^c$ is simply the Gaussian process prior. Namely,

$$p(\mathbf{u}^c) = \mathcal{N}\left(\mathbf{u}^c|\mathbf{0}, \mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c\right).$$

Importantly, now one only has to invert the matrices $\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c$ of size $M \times M$ and compute the product $\mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}$. Therefore, the training cost will be in $\mathcal{O}(M^2 N)$, which is significantly better if $M \ll N$.

In practice, the inducing point values $\mathbf{u}^c$ will be unknown and they are treated as latent variables of the model. An approximate Gaussian posterior distribution will be specified for them. Namely, $q(\mathbf{u}^c) = \mathcal{N}(\mathbf{u}^c|\mathbf{m}_c, \mathbf{S}_c)$ for $c = 1, \ldots, C$. This uncertainty about $\mathbf{u}^c$ can be readily introduced in (4) simply by marginalizing these latent variables. The result is a Gaussian distribution for $\mathbf{f}^c$ with extra variance due to the randomness of $\mathbf{u}^c$. That is,

$$p(\mathbf{f}^c|\mathbf{y}) \approx \int p(\mathbf{f}^c|\mathbf{u}^c)q(\mathbf{u}^c) = \mathcal{N}(\mathbf{f}^c|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \qquad (5)$$

where

$$\boldsymbol{\mu}_c = \mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{m}_c,$$
$$\boldsymbol{\Sigma}_c = \mathbf{K}_{\mathbf{X},\mathbf{X}}^c - \mathbf{K}_{\mathbf{X},\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c + \mathbf{S}_c)(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{K}_{\mathbf{Z}^c,\mathbf{X}}^c.$$

In the following sections we describe how to compute the parameters of each approximate distribution $q(\mathbf{u}^c)$, for $c = 1, \ldots, C$, using variational inference.

3. Multi-class GP Classification with Input Noise

In this section we describe the proposed approaches for dealing with noisy inputs in the context of multi-class GP classification. Let us consider that $\tilde{\mathbf{X}}$ is the matrix of noisy observations in which the data patterns are contaminated with additive Gaussian noise

with some mean and some variance. Again, consider that $\mathbf{X}$ is the matrix of noiseless inputs. That is,

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i + \boldsymbol{\epsilon}_i \,, \qquad\qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{V}_i) \,, \qquad\qquad (6)$$

where $\tilde{\mathbf{x}}_i \in \mathbb{R}^d$ is a particular observation and $\mathbf{V}_i$ is a $d \times d$ diagonal matrix. That is, we assume independent additive Gaussian noise for the inputs. For the moment, we consider that the variance of the additive noise $\mathbf{V}_i$ is known beforehand. Later on, we describe how this parameter can be inferred from the training data. Recall that we assume input attributes in the real line.

3.1 Modeling the Input Noise Using Latent Variables

A first approach for taking into account noisy inputs $\tilde{\mathbf{X}}$ in the context of GP multi-class classification is based on making approximate Bayesian inference about the actual noiseless inputs $\mathbf{x}_i$, for $i = 1, \ldots, N$. Importantly, these variables will be latent. The observed variables will be the ones contaminated with Gaussian noise $\tilde{\mathbf{x}}_i$, for $i = 1, \ldots, N$. With this goal, note that the assumption made in (6) about the generation of $\tilde{\mathbf{x}}_i$ provides a likelihood function for the actual observation. That is,

$$p(\tilde{\mathbf{x}}_i | \mathbf{x}_i) = \mathcal{N}(\tilde{\mathbf{x}}_i | \mathbf{x}_i, \mathbf{V}_i) \,.$$

In order to make inference about the noiseless observation $\mathbf{x}_i$, we need to specify a prior distribution for that variable. In practice, however, the actual prior distribution for $\mathbf{x}_i$ is specific of each classification problem and unknown, in general. Therefore, we set the prior for $\mathbf{x}_i$ to be a multi-variate Gaussian with a broad variance. Namely,

$$p(\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_i | \mathbf{0}, \mathbf{I}s) \,,$$

where $\mathbf{I}$ is the identity matrix and s is chosen to have a large value, *i.e.*, $s = 1,000$, in order to make it similar to a non-informative uniform distribution. This prior has shown good results in our experiments.

3.1.1 JOINT AND POSTERIOR DISTRIBUTION

The first step towards making inference about $\mathbf{x}_i$ is to describe the joint distribution of all the variables of the model (observed and latent). This distribution is given by

$$p(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{y}, \mathbf{F}, \mathbf{U}) = \left[\prod_{i=1}^{N} p(y_i | \mathbf{f}_i) \right] \left[\prod_{c=1}^{C} p(\mathbf{f}^c | \mathbf{u}^c) p(\mathbf{u}^c) \right] \left[\prod_{i=1}^{N} p(\tilde{\mathbf{x}}_i | \mathbf{x}_i) p(\mathbf{x}_i) \right] \,,$$

where $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N)^{\mathrm{T}}$ is the matrix of noisy observations, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^{\mathrm{T}}$ is the matrix of actual noiseless inputs, $\mathbf{y}$ is the vector of observed labels, $\mathbf{F} = (\mathbf{f}_1, \ldots, \mathbf{f}_N)^{\mathrm{T}}$ is the matrix of process values at the actual noiseless inputs (*i.e.*, at each $\mathbf{x}_i$ instead of at each $\tilde{\mathbf{x}}_i$), and $\mathbf{U} = (\mathbf{u}^1, \ldots, \mathbf{u}^C)^{\mathrm{T}}$ is the matrix of process values at the inducing points.

The posterior distribution of the latent variables, *i.e.*, $\mathbf{X}$, $\mathbf{F}$ and $\mathbf{U}$ is obtained using Bayes' rule:

$$p(\mathbf{X}, \mathbf{F}, \mathbf{U} | \mathbf{y}, \tilde{\mathbf{X}}) = \frac{p(\tilde{\mathbf{X}}, \mathbf{X}, \mathbf{y}, \mathbf{F}, \mathbf{U})}{p(\mathbf{y}, \tilde{\mathbf{X}})} \,. \qquad\qquad (7)$$

Again, as in the case of standard multi-class classification where there is noise in the inputs, computing this posterior distribution is intractable and approximate inference will be required. To approximate this distribution we employ variational inference, as described in the next section.

3.1.2 Approximate Inference Using Variational Inference

We will use variational inference (VI) as the approximate inference method (Jordan et al., 1999). The posterior approximation that will target the exact posterior (7) is specified to be

$$q(\mathbf{X}, \mathbf{F}, \mathbf{U}) = \left[\prod_{c=1}^{C} p(\mathbf{f}^c | \mathbf{u}^c) q(\mathbf{u}^c) \right] \left[\prod_{i=1}^{N} q(\mathbf{x}_i) \right], \tag{8}$$

where

$$q(\mathbf{u}^c) = \mathcal{N}(\mathbf{u}^c | \mathbf{m}_c, \mathbf{S}_c), \qquad\qquad q(\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_i^x, \mathbf{V}_i^x), \tag{9}$$

with $\mathbf{V}_i^x$ a diagonal matrix. This posterior approximation assumes independence among the different GPs of the model and the actual noiseless inputs $\mathbf{X}$.

To enforce that q looks similar to the target posterior distribution, VI minimizes the Kullback-Leibler (KL) divergence between q and the exact posterior p, given by the distribution in (7). This is done indirectly by maximizing the evidence lower bound $\mathcal{L}$. See Jordan et al. (1999) for further details. The evidence lower bound (ELBO) is given by

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_q \left[\log \frac{p(\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{y}, \mathbf{F}, \mathbf{U})}{q(\mathbf{X}, \mathbf{F}, \mathbf{U})} \right] \\
&= \sum_{i=1}^{N} \mathbb{E}_q \left[\log p(y_i | \mathbf{f}_i) \right] + \sum_{i=1}^{N} \mathbb{E}_q [\log p(\tilde{\mathbf{x}}_i | \mathbf{x}_i)] \\
&\quad - \sum_{c=1}^{C} \mathrm{KL}(q(\mathbf{u}^c) | p(\mathbf{u}^c)) - \sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{x}_i) | p(\mathbf{x}_i)),
\end{aligned} \tag{10}$$

where $\mathrm{KL}(\cdot | \cdot)$ is the Kullback-Leibler divergence and where we have used the fact that the factors of the form $p(\mathbf{f}^c | \mathbf{u}^c)$ described in (4) and present in both the joint distribution and in q will cancel.

One problem that arises when computing the previous expression is that the first expectation in (10), *i.e.* $\mathbb{E}_q \left[\log p(y_i | \mathbf{f}_i) \right]$, does not have a closed form solution. It can, however, be computed using a one dimensional quadrature and Monte Carlo methods combined with the reparametrization trick (Kingma and Welling, 2014; Hensman et al., 2015b). Concerning the other factors, the second expectation, $\sum_{i=1}^{N} \mathbb{E}_q[\log p(\mathbf{x}_i | \tilde{\mathbf{x}}_i)]$, is the expectation of the logarithm of a Gaussian distribution, so it can be computed in a closed form. We can also evaluate analytically the KL divergences in the lower bound, $\sum_{k=1}^{K} \mathrm{KL}(q(\mathbf{u}^k) | p(\mathbf{u}^k)) - \sum_{i=1}^{N} \mathrm{KL}(q(\tilde{\mathbf{x}}_i) | p(\tilde{\mathbf{x}}_i))$, since they involve Gaussian distributions.

Importantly, the ELBO in (10) is expressed as a sum across the observed data points. This means that this objective is suitable for being optimized using mini-batches. The

9

required gradients can be obtained using automatic differentiation techniques such as those implemented in frameworks such as Tensorflow (Abadi et al., 2015). Appendix A describes the details about how to obtain an unbiased noisy estimate of the ELBO, $\mathcal{L}$.

One last remark is that it is possible to show that

$$\log p(\mathbf{y}, \tilde{\mathbf{X}}) = \mathcal{L} + \text{KL}[q||p]\,,$$

where $\text{KL}[q||p]$ is the Kullback-Leibler (KL) divergence between q and the target posterior distribution p in (7) (Jordan et al., 1999). After maximizing the ELBO, $\mathcal{L}$, it is expected that the KL term is fairly small and hence $\log p(\mathbf{y}, \tilde{\mathbf{X}}) \approx \mathcal{L}$. Therefore, $\mathcal{L}$ can be maximized to find good values for the model hyper-parameters. This is expected to maximize $\log p(\mathbf{y}, \tilde{\mathbf{X}})$, which will correspond to a type-II maximum likelihood approach (Rasmussen and Williams, 2006). The locations of the inducing points $\mathbf{Z}^c$, for $c = 1, \ldots, C$, are found by maximizing $\mathcal{L}$, an estimate of the marginal likelihood, as in Hensman et al. (2015b); Villacampa-Calvo and Hernández-Lobato (2017). Note, however, that these correspond to parameters of the posterior approximation q, defined in (8), and not of the described probabilistic model (Titsias, 2009).

3.1.3 PREDICTIONS

After the maximization of the lower bound (10), the approximate distribution q is fitted to the actual posterior. The predictive distribution for the class label $y_\star$ of a new instance $\mathbf{x}_\star$ can be approximated by replacing the exact posterior by the posterior approximation in the exact predictive distribution. Namely,

$$p(y_\star|\mathbf{x}_\star) \approx \int p(y_\star|\mathbf{f}_\star) \left[\prod_{c=1}^{C} p(f_\star^c|\mathbf{u}^c) q(\mathbf{u}^c) d\mathbf{u}^c \right] p(\mathbf{x}_\star|\tilde{\mathbf{x}}_\star) d\mathbf{x}_\star d\mathbf{f}_\star\,, \tag{11}$$

where $p(\mathbf{x}_\star|\tilde{\mathbf{x}}_\star)$ is the posterior distribution of the actual attributes of the new instance given the observed attributes $\tilde{\mathbf{x}}_\star$. This posterior is the normalized product of the prior times the likelihood. Therefore, it can be computed in closed form and is a Gaussian. Namely,

$$p(\mathbf{x}_\star|\tilde{\mathbf{x}}_\star) = \frac{p(\tilde{\mathbf{x}}_\star|\mathbf{x}_\star)p(\mathbf{x}_\star)}{p(\tilde{\mathbf{x}}_\star)} = \mathcal{N}(\mathbf{x}_\star|\boldsymbol{\mu}_\star^x, \mathbf{V}_\star^x)\,,$$

where $\mathbf{V}_\star^x = (\mathbf{V}_\star^{-1} + \mathbf{I}s^{-1})^{-1}$ and $\boldsymbol{\mu}_\star^x = \mathbf{V}_\star^x(\mathbf{V}_\star^{-1}\tilde{\mathbf{x}}_\star)$, and where $\mathbf{V}_\star$ is a diagonal matrix with the variances of the Gaussian noise around $\mathbf{x}_\star$. Note that if s is fairly large, as it is in our case, then essentially $\mathbf{V}_\star^x \approx \mathbf{V}_\star$

In general, the integral in (11) is intractable. However, we can generate samples of $\mathbf{x}_\star$ simply by drawing from $p(\mathbf{x}_\star|\tilde{\mathbf{x}}_\star)$ to then compute a Monte Carlo approximation:

$$p(y_\star|\mathbf{x}_\star) \approx \frac{1}{S} \sum_{s=1}^{S} \int p(y_\star|\mathbf{f}_\star^s) \left[\prod_{c=1}^{C} p(f_{s,\star}^c|\mathbf{u}^c) q(\mathbf{u}^c) d\mathbf{u}^c \right] d\mathbf{f}_\star^s\,, \tag{12}$$

where S is the number of samples, $\mathbf{f}_\star^s = (f^1(\mathbf{x}_\star^s), \ldots, f^C(\mathbf{x}_\star^s))^\text{T}$, $f_{s,\star}^c = f^c(\mathbf{x}_\star^s)$ with $\mathbf{x}_\star^s$ the generated s sample of $\mathbf{x}_\star$.

The only remaining thing is how to compute the integral in the right hand side of (12). It turns out that the integral with respect to each $\mathbf{u}^c$ can be computed analytically using (5). The integral with respect to $\mathbf{f}_\star^s$ can be approximated using a one-dimensional quadrature. In particular, under the likelihood function in (2), the approximation of the predictive distribution becomes

$$p(y_\star|\mathbf{x}_\star) \approx \frac{\epsilon}{C-1} + (1-\epsilon)\frac{1}{S}\sum_{s=1}^{S}\int \mathcal{N}(f_s^{y_\star}|m_{y_\star}^s, v_{y_\star}^s)\prod_{c\neq y_\star}\Phi\left(\frac{f_s^c - m_c^s}{\sqrt{v_c^s}}\right)df_s^{y_\star}, \qquad (13)$$

where $f_s^{y_\star} = f^{y_\star}(\mathbf{x}_\star^s)$, $f_s^c = f^c(\mathbf{x}_\star^s)$, $\Phi(\cdot)$ is the cumulative probability of a standard Gaussian distribution and

$$m_c^s = \mathbf{k}_{\mathbf{x}_\star^s,\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{m}_c\,,$$
$$v_c^s = k_{\mathbf{x}_\star^s,\mathbf{x}_\star^s}^c - \mathbf{k}_{\mathbf{x}_\star^s,\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{K}_{\mathbf{Z}^c,\mathbf{x}_\star^s} + \mathbf{k}_{\mathbf{x}_\star^s,\mathbf{Z}^c}^c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{S}_c(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{k}_{\mathbf{Z}^c,\mathbf{x}_\star^s}^c\,,$$

for $c = 1, \ldots, C$ with $k_{\mathbf{x}_\star^s,\mathbf{x}_\star^s}^c$ the variance of $f^c(\cdot)$ at $\mathbf{x}_\star$, $\mathbf{k}_{\mathbf{x}_\star^s,\mathbf{Z}^c}^c$ the matrix of covariances between the values of $f^c(\cdot)$ at $\mathbf{x}_\star^s$ and $\mathbf{Z}^c$, and $\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c$ the covariances of $f^c(\cdot)$ among the inducing points $\mathbf{Z}^c$. Note that the integral over $f_s^{y_\star}$ in (13) has no closed form solution but it can be computed using one-dimensional quadrature.

3.2 Amortized Approximate Inference

A limitation of the method described is that the approximate posterior distribution over the noiseless inputs, $i.e.$, $\prod_{i=1}^{N}q(\mathbf{x}_i)$, demands storing in memory a number of parameters that is in $\mathcal{O}(N)$, where N is the number of observed instances. Of course, in the big data regime, $i.e.$, when N is in the order of thousands or even millions the memory resources needed to store those parameters can be too high. To alleviate this problem, we propose to use amortized approximate inference to reduce the number of parameters that need to be store in memory (Kingma and Welling, 2014; Shu et al., 2018).

Amortized variational inference assumes that the parameters of each distribution $q(\mathbf{x}_i)$, $i.e.$, the mean and the diagonal covariance matrix, can be obtained simply as a non-linear function that depends on the observed data instance $(\tilde{\mathbf{x}}_i, y_i)$. This non-linear function is set to be a neural network whose parameters are optimized during training. That is,

$$q(\mathbf{x}_i) = \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_i, y_i), \mathbf{V}_\theta(\tilde{\mathbf{x}}_i, y_i))\,,$$

where both $\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_i, y_i)$ and $\mathbf{V}_\theta(\tilde{\mathbf{x}}_i, y_i)$ are obtained as the output of a neural network with parameters θ (we use a one-hot encoding for the label y_i). Therefore, one only has to store in memory the neural network which has a fixed number of parameters. This number of parameters does not depend on N. The neural network can be adjusted simply by maximizing w.r.t θ the evidence lower bound described in (10). The computational cost of the method is not changed, since the cost for the feed-forward pass of the network to obtain $\boldsymbol{\mu}_\theta(\tilde{\mathbf{x}}_i, y_i)$ and $\mathbf{V}_\theta(\tilde{\mathbf{x}}_i, y_i)$ is constant.

Amortized approximate inference introduces the inductive bias that points that are located in similar regions of the input space should have similar parameters in the corresponding posterior approximation $q(\mathbf{x}_i)$. Of course, this has the benefit property of reducing the

number of parameters of the approximate distribution q. A second advantage is, however, that the neural network can provide a beneficial regularization that is eventually translated into better generalization results (Shu et al., 2018). More precisely, our experiments show that amortized variational inference sometimes provides better results than using an approximate distribution q that has separate parameters per each data point $\tilde{\mathbf{x}}_i$.

Besides using this neural network to compute the parameters of $q(\mathbf{x}_i)$, the model is not changed significantly. Prediction is done in the same way, and hyper-parameter optimization is also carried out by maximizing the evidence lower bound.

3.3 First Order Approximation

In this section we describe an alternative method to account for input noise in the context of multi-class classification with GPs. This method is inspired on work already done for regression problems (Mchutchon and Rasmussen, 2011). Consider the relation between the noisy and the noiseless input measurements given by (6). Now, consider a Taylor expansion of a latent function $f(\cdot)$ around the noiseless measurement $\mathbf{x}_i$. Namely,

$$f(\mathbf{x}_i + \boldsymbol{\epsilon}_i) = f(\mathbf{x}_i) + \boldsymbol{\epsilon}_i^{\mathrm{T}} \frac{\partial f(\mathbf{x}_i)}{\partial \mathbf{x}_i} + \cdots \approx f(\tilde{\mathbf{x}}_i) + \boldsymbol{\epsilon}_i^{\mathrm{T}} \frac{\partial f(\tilde{\mathbf{x}}_i)}{\partial \tilde{\mathbf{x}}_i} + \cdots \tag{14}$$

where $\tilde{\mathbf{x}}_i$ is the noisy observation. Since we do not have access to the noiseless input vector $\mathbf{x}_i$, we simply approximate it with the noisy one $\tilde{\mathbf{x}}_i$. Note that this last expression involves the derivatives of the GP. Although they can be showed to be again GPs (see Rasmussen and Williams (2006) for further details), we here decide to approximate these derivatives with the derivatives of the mean of GP, as in Mchutchon and Rasmussen (2011). This approximation corresponds to ignoring the uncertainty about the derivative. Let $\partial_{\overline{f}}(\tilde{\mathbf{x}}_i)$ denote the d-dimensional vector corresponding to the derivative of the mean of the GP with respect to each input dimension at $\tilde{\mathbf{x}}_i$. If we expand the right hand side of (14) up to the first order terms, we get a linear model. Namely,

$$f(\mathbf{x}_i + \boldsymbol{\epsilon}_i) = f(\tilde{\mathbf{x}}_i) + \boldsymbol{\epsilon}_i^{\mathrm{T}} \partial_{\overline{f}}(\tilde{\mathbf{x}}_i) \,.$$

Therefore, the input noise can be understood as output noise whose variance is proportional to square of the derivative of the mean value of f at $\tilde{\mathbf{x}}_i$.

The model just described can be combined with the framework for sparse GPs described in Section 2.1 to give an alternative posterior predictive distribution for $\mathbf{f}^c$ to the one described in (5). That is,

$$p(\mathbf{f}^c|\mathbf{u}^c) = \mathcal{N}\left(\mathbf{f}^c|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right) , \tag{15}$$

with

$$\boldsymbol{\mu}_c = \mathbf{K}_{\tilde{\mathbf{X}},\mathbf{Z}^c}^c (\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1} \mathbf{m}_c \,,$$

$$\boldsymbol{\Sigma}_c = \mathbf{K}_{\tilde{\mathbf{X}},\tilde{\mathbf{X}}}^c - \mathbf{K}_{\tilde{\mathbf{X}},\mathbf{Z}^c}^c (\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1} \left((\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1} - \mathbf{S}_c\right) (\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1} \mathbf{K}_{\mathbf{Z}^c,\tilde{\mathbf{X}}} + \boldsymbol{\Delta} \,,$$

where $\mathbf{m}_c$ and $\mathbf{S}_c$ are the parameters of $q(\mathbf{u}^c)$, the covariance matrices are evaluated at the noisy measurements $\tilde{\mathbf{X}}$ and

$$\Delta_{i,i} = \partial_{\overline{f}^c}(\tilde{\mathbf{x}}_i)^{\mathrm{T}} \mathbf{V}_i \partial_{\overline{f}^c}(\tilde{\mathbf{x}}_i) \,. \tag{16}$$

Therefore, $\boldsymbol{\Delta}$ is a diagonal matrix whose entries account for the extra output noise that results from the corresponding input noise. This makes sense, since the input noise is expected to have a small effect in those regions of the input space in which the latent function is expected to be constant. Importantly, in the sparse setting

$$\partial_{\overline{f}^c}(\tilde{\mathbf{x}}_i) = \frac{\partial \boldsymbol{\mu}_c}{\partial \tilde{\mathbf{x}}_i} = \frac{\partial \mathbf{k}^c_{\tilde{\mathbf{x}}_i, \mathbf{Z}^c}(\mathbf{K}^c_{\mathbf{Z}^c, \mathbf{Z}^c})^{-1} \mathbf{m}_c}{\partial \tilde{\mathbf{x}}_i} \, .$$

This partial derivative can be easily obtained automatically in modern frameworks for implementing multi-class GP classifiers such as Tensorflow (Abadi et al., 2015). The expression in (15) can replace (5) in a standard sparse multi-class GP classifier to account for input noise. One only has to provide the corresponding input noise variances $\mathbf{V}_i$, for $i = 1, \ldots, N$. Approximate inference in such a model can be carried out using variational inference, as described in Hensman et al. (2015b).

Figure 1 shows the predictive distribution of the model described, which we refer to as NIMGP$_{\text{FO}}$, for each latent function in a three class toy classification problem described in Section 5.1. The figure on the left shows the predictive distribution obtained when the extra term in the predictive variance $\boldsymbol{\Delta}$ that depends on the slope of the mean is ignored in (15). The figure on the right shows the resulting predictive distribution when that term is taken into account. We can observe that the produced effect is to increase the variance of the predictive distribution by some amount that is proportional to the squared value of slope of the predictive mean. This is particularly noticeable in the case of the latent function corresponding to class number 2. A bigger variance in the predictive distribution for the latent function will correspond to less confidence in the predictive distribution for the class label. See Section 5.1 for further details.

Figure 1 also shows the learned locations of the inducing points for each latent function (displayed at the bottom of each image). We observe that they tend to be placed uniformly in the case of the latent functions corresponding to class labels 0 and 1. However, in the case of the latent function corresponding to class label 2, they concentrate in specific regions of the input space. Namely, in those regions in which the latent function changes abruptly.

The inductive bias of a machine learning algorithm is a set of assumptions that the chosen methodology uses to predict outputs given, in this case, uncertain inputs. The stochastic estimate of the lower bound, in contrast w.r.t the linear approximation of NIMGP$_{\text{FO}}$, is unbiased, as it is shown in Appendix A. In the NIMGP$_{\text{FO}}$ method, the input noise is converted to output noise through a linear approximation based in a second-order Taylor expansion, that assumes that the input noise distribution fits a quadratic approximation and regularity assumptions that need not be necessarily true.

3.4 Learning the Level of Noise in the Inputs

The previous sections assumed that the variance of Gaussian noise associated to each input dimension, *i.e.*, the diagonal matrix $\mathbf{V}_i$, is known before-hand. This is the case of many practical problems in which the error associated to the measurement instrument that is used to obtain the observed attributes $\tilde{\mathbf{x}}_i$ is well-known. However, in certain situations it can be the case that the variance of the error is unknown. In this case, it may still be possible to infer this level of error from the observed data.
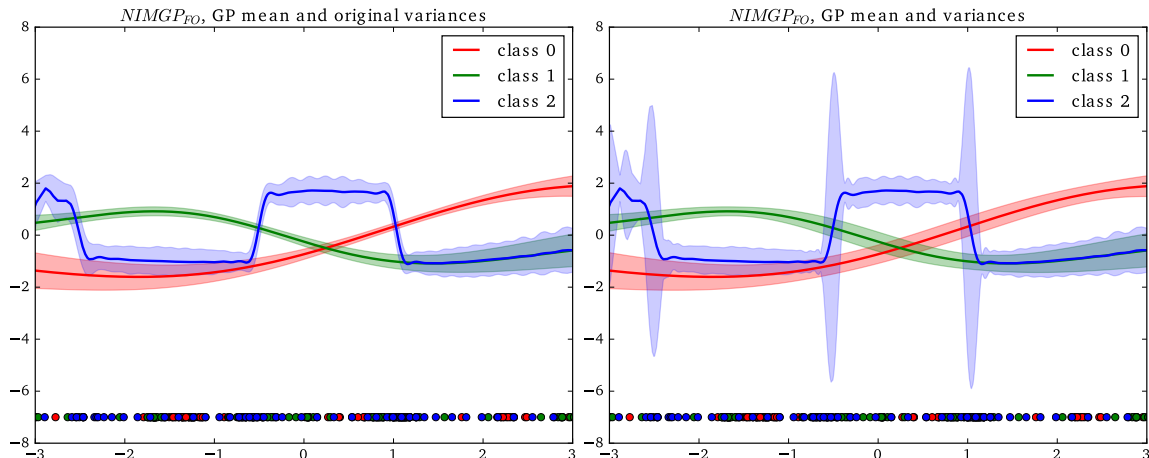
Figure 1: For the NIMGP$_{\text{FO}}$ model, the predictive mean GP and variances (right) and original variances (left), for each latent function, for a classification problem with three classes shown in red, green and blue. The learned locations of the inducing points, $\mathbf{Z}^c$, are shown in a row at the bottom of each figure. Best seen in color.

Typically, in this case, one will assume that the level of noise is the same across all observed data instances. That is, $\mathbf{V}_i = \mathbf{V}$. This has the advantage of reducing the number of parameters that have to be inferred. To estimate $\mathbf{V}$ one can simply treat this parameter as a hyper-parameter of the model. Its value can be estimated simply by type-II maximum likelihood, as any other hyper-parameter of the GP (Rasmussen and Williams, 2006). Under this setting, one simply maximizes the marginal likelihood of the model, *i.e.*, the denominator in Bayes' theorem w.r.t the parameter of interest. This is precisely the approach followed in Mchutchon and Rasmussen (2011) for regression problems.

Because evaluating the marginal likelihood is infeasible in the models described so far, one has to use an approximation. This approximation can be the evidence lower bound described in (10), which will be similar to the marginal likelihood if the approximate distribution q is an accurate posterior approximation. The maximization of (10) w.r.t $\mathbf{V}$ can be simply done with no extra computational cost using again stochastic optimization techniques.

In general, we will assume that $\mathbf{V}_i$ is known for each data instance. If that is not the case, we will infer that level of noise using the method described in this section.

3.5 Other Problems Different from Multi-class Classification

The methods described so far focus on multi-class problems as we are initially motivated by a real-world application from the field of astrophysics. This setup is challenging since there is a latent function for each potential class label. This also results in complex likelihood factors and the need to approximate the first expectation in (10) using one dimensional quadrature and Monte Carlo methods. Of course, addressing regression problems is straight-forward. In this case, the likelihood factors are Gaussian and some of the required computations to

evaluate the (10) are tractable (the only expectation that needs to be approximated is that with respect to each $q(\mathbf{x}_i)$). A single GP is used in this case. There has been plenty of research work considering this setting in the literature and we refer the reader to Section 4.

We describe how the proposed method can be used in problems involving other non-Gaussian likelihood functions. For example, in problems in which the task of interest is to predict count data, we may want to use the likelihood factors that are Poisson distributed (Diggle et al., 1998). That is, $p(y_i|f_i) = \text{Pois}(y_i|\lambda_i)$, where λ_i is the rate parameter that may non-linearly depend on $\mathbf{x}_i$ through a GP, $e.g.$, $\lambda_i = \exp(f_i)$, with f_i the process value at $\mathbf{x}_i$. Similarly, in binary classification, where $y_i \in \{-1, 1\}$, the likelihood factors can be of the form $p(y_i|f_i) = \Phi(f_iy_i)$, where $\Phi(\cdot)$ is the sigmoid or the probit function (Rasmussen and Williams, 2006). Multi-label data can also be considered by using similar sigmoid likelihood factors, several GPs and a linear transformation to account for each potential label (Panos, 2019). A robustified GP for regression can be considered by using Student-t factors in the likelihood, as an alternative to the standard Gaussian factors (Neal, 1997). Namely, $p(y_i|f_i) = \text{Student}(y_i|f_i, \sigma, \mu)$, where f_i, σ and ν are the location, scale and degrees of freedom parameters, respectively. Other problems may involve ordinal regression, where the target variables are discrete but ordered (Chu and Ghahramani, 2005). In that case the likelihood function may involve two probit functions to account for the fact that the process value f_i associated to $\mathbf{x}_i$ should lie in a particular interval of the real line.

All these problems are characterized by non-Gaussian likelihood factors involving a single GP. In this case, the expectation of $\mathbb{E}_q[\log p(y_i|f_i)]$ with respect to $q(f_i)$ required to evaluate (10) is intractable. However, it can be easily approximated also using a one-dimensional quadrature. The expectation with respect to each $q(\mathbf{x}_i)$ can be approximated using Monte Carlo methods, as in the case of multi-class classification. The evaluation of the proposed method in the described settings is left for future work since the focus of this paper is multi-class classification.

3.6 Summary of the Proposed Methods to Deal with Input Noise

Below we briefly describe the different methods explained so far to deal with input noise in the context of multi-class GP classification:

- **NIMGP**: This is the method described in Section 3.1 which uses latent variables to model the original noiseless inputs. It relies on a Gaussian distribution for the actual observation with the mean being a random variable representing the noiseless input. We assume a non-informative Gaussian prior for the noiseless input. The joint posterior distribution that involves non-Gaussian likelihood factors for the process values is approximated using variational inference. Quadrature and Monte Carlo methods are combined with the reparametrization trick to obtain a noisy estimate of the lower bound which is optimized using stochastic methods.

- **NIMGP$_{\mathbf{NN}}$:** A limitation of the previous method is the need of storing parameters for each data instance. This is a disadvantage in big data applications. To circumvent this problem, amortized approximate inference is employed in this method, as explained in Section 3.2. We use a neural network to compute the parameters of posterior approximation for the noiseless attributes of each data instance. This network reduces

the number of parameters of the approximate distribution and also regularizes the model. Approximate inference is carried out as in NIMGP.

- **NIMGP$_{\mathbf{FO}}$**: This is the method described in Section 3.3. In this method we adapt for classification with sparse GPs an already proposed method for regression that accounts for noisy inputs using GPs. This method is based on propagating the noise found in the inputs to the variance of the GPs predictive distribution. For this, a local linear approximation of the GP is used at each input location. This allows the input noise to be recast as output noise proportional to the squared gradient of the GP predictive mean.

A summary of the parameters employed by each method is provided in Appendix C.

4. Related Work

In the literature there are some works dealing with GPs and input points corrupted with noise in the context of regression problems (Mchutchon and Rasmussen, 2011). For this, a local linear approximation of the GP at each input point is performed. When this is done, the input noise is translated into output noise proportional to the squared gradient of the GP posterior mean. Therefore, the mentioned paper simplifies the problem of modeling input noise by assuming that the input measurements are deterministic and by inflating the corresponding output variance to compensate for the extra noise. When this operation is performed, it leads to output noise variance varying across the input space. This property is defined as *heteroscedasticity*. The model presented in that paper can hence be described as a heteroscedastic GP model. We have extended the linearization approach of Mchutchon and Rasmussen (2011) to address multi-class classification problems as they only considered regression problems. Furthermore, we have compared such a method, NIMGP$_{FO}$, with another approach that uses latent variables to model the noiseless inputs, NIMGP, and that, in principle, does not rely on a linear approximation of the GP. We show that on synthetic data NIMGP performs better than NIMGP$_{FO}$ which indicates that the linear approximation used in NIMGP$_{FO}$ may be inaccurate. In fact, the GP is non-linear in practice. Note that we also consider sparse GPs instead of standard GPs, which make our approach more scalable to data sets with a larger number of instances. These differences are common between our proposed methods and most of the techniques described in this section.

As described in the previous paragraph, one of our proposed methods can be understood as a GP model in which the level of output noise depends on the input location, *i.e.*, the data can be considered to be heteroscedastic. Several works have tried to address such a problem in the context of GPs and regression tasks. In particular, Goldberg et al. (1998) introduce a second GP to deal with the output noise level as a function of the input location. This approach uses a Markov chain Monte Carlo method to approximate the posterior noise variance, which is time-consuming. A more efficient MAP estimation approach is described by Le et al. (2005). This other method is applicable in models with likelihood factors in the exponential family and it leads to a convex optimization problem solvable by the Newton method. An interesting extension to the work of Goldberg et al. (1998) tries to circumvent its computational limitations by replacing the Monte Carlo method with an approximative

most likely noise approach (Kersting et al., 2007). This other work learns both the hidden noise variances and the kernel parameters simultaneously. Furthermore, it significantly reduces the computational cost.

Other related work concerning heteroscedasticity in the context of regression problems is the one of Lázaro-Gredilla and Titsias (2011). This approach also relies on variational inference for approximate inference in the context of GPs. These authors explicitly take into account the input noise, and model it with GP priors. By using an exponential transformation of the noise process, they specify the variance of the output noise. Exact inference in the heteroscedastic GP is intractable and the computations need to be approximated using variational inference. Variational inference in such a model has equivalent cost to an analytically tractable homoscedastic GP. Importantly, this work focuses exclusively on regression and ignores classification tasks. Furthermore, no GP sparse approximation is considered by these authors. Therefore, the problems addressed cannot contain more than a few thousand instances.

Copula processes are another alternative to deal with input noise in the context of regression tasks and GPs (Wilson and Ghahramani, 2010). In this case, approximate inference is carried out using the Laplace approximation and Markov chain Monte Carlo methods. There also exists in the literature an approach for online heteroscedastic GP regression that tackles the incorporation of new measurements in constant run-time and makes the computation cheaper by considering online sparse GPs (Bijl et al., 2017). This approach has proven to be effective in a practical applications considering, *e.g.*, system identification.

The work of Mchutchon and Rasmussen (2011) has been employed in several practical applications involving machine learning regression problems. For example, in a problem concerning driving assistant systems (Armand et al., 2013), where the velocity profile that the driver follows is modeled as the vehicle decelerates towards a stop intersection. Another example application can be found in the context of Bayesian optimization (Nogueira et al., 2016; Oliveira et al., 2017), where a GP models an objective function that is being optimized. In this scenario, the input space is contaminated with i.i.d Gaussian noise. Two real applications are considered: safe robot grasping and safe navigation under localization uncertainty (Nogueira et al., 2016; Oliveira et al., 2017).

Another approach that considers input noise in the context of regression problems in arbitrary models is that of Bócsi and Csató (2013). This approach corrects the bias caused by the integration of the noise. The correction is proportional to the Hessian of the learned model and to the variance of the input noise. The advantage of the method is that it works for arbitrary regression models and the disadvantage is that it does not improve prediction for high-dimensional problems, where the data are implicitly scarce, and the estimated Hessian is considerably flattened.

Input dependent noise has also been taken into account in the context of binary classification with GPs by Hernández-lobato et al. (2014). In particular, these authors describe the use of an extra GP to model the variance of additive Gaussian noise around a latent function that is used for binary classification. This latent function is modeled again using a GP. Importantly, in this work the level of noise is expected to depend on privileged information. These are extra input attributes that are only available at training time, but not at test time. The goal is to exploit that privileged information to obtain a better classifier during the training phase. Approximate inference is done in this case using expectation

propagation instead of variational inference (Minka, 2001). The experiments carried out show that privileged information is indeed useful to obtain better classifiers based on GPs.

Importantly, in all these related works involving output noise that depends on the input, the domain of application is different from ours since only regression or binary classification problems are addressed. Furthermore, it is assumed a level of correlation between the location of the input features and the amount of extra output noise. This relation between input features and the extra output noise can be arbitrarily complicated, since it is most of the times modeled via another GP. In NIMGP_{FO} the level of extra output noise not only depends on the input location of the instance, but also on the variance of the input noise associated to that particular instance. Moreover, the expression for the dependence of the output noise with respect to the input is fixed (it cannot be learned from the data) and is given by the second derivative of the GP. See Eq. (16) for further details. This expression is just obtained as a consequence of modeling in an approximate way the input noise.

Random inputs have also been considered in the context of GPs for un-supervised learning in the Bayesian GP latent variable model (GP-LVM) (Titsias and Lawrence, 2010; Damianou et al., 2016). In this model the observed data is assumed to be obtained using a non-linear mapping applied to a set of latent variables. The non-linear mapping is modeled using GPs and the latent variables are estimated using a Bayesian approach. This involves setting a prior distribution for their potential values and using variational inference (VI) to compute an approximate posterior distribution. This framework is very similar to the VI approach we describe for the proposed method NIMGP. The main difference, however, between NIMGP and the Bayesian GP-LVM is that the latter can only address unsupervised learning problems and it requires tractable Gaussian likelihood factors. A generalization of the Bayesian GP-LVM to account for partially observed input attributes is described by Damianou and Lawrence (2015). The resulting framework also resembles the VI approach described for the NIMGP method. However, in that case, the approximate distribution q over the latent attributes is constrained to be a Gaussian with fixed mean and adjustable variance, unlike the approximate distribution in Eq. (9), which includes adjustable means. Again, this model requires tractable Gaussian likelihood factors and cannot be used to address multi-class classification problems, like NIMGP.

A generalization of the Bayesian GP-LVM considers a concatenation of non-linear transformations of the latent features using GPs. This leads to deep GPs for unsupervised learning (Damianou and Lawrence, 2013). The applicability of such a method is, however, restricted to small data sets partially because of the large number of variational parameters that grows linearly with the size of the data set. This problem can be alleviated by augmenting the model with a recognition system that constraints the variational posterior over the latent variables of each layer (Dai et al., 2016). The recognition system is given by a neural network that amortizes the approximate distribution q, as in the proposed method NIMGP_{NN}. The main difference is that, although the extension to address regression problems is straight-forward, it is not clear how to use such a method for classification problems.

To tackle input noise in the context multi-class classification, when one does does not rely on the use of GPs, some decomposition strategies can be used. Specifically, it is possible to decompose the problem into several binary classification subproblems, reducing the complexity and, hence, dividing the effects caused by the noise into each of the subproblems

(Sáez et al., 2014). There exist several of these decomposition strategies, being the one-vs-one scheme a method that can be applied for well known binary classification algorithms. The results obtained by these authors show that the one-vs-one decomposition leads to better performances and more robust classifiers than other decompositions. A problem of these decompositions, however, is that they can lead to ambiguous regions in which it is not clear what class label should be predicted (Bishop, 2006). Furthermore, they do not learn an underlying true multi-class classifier and rely on binary classifiers to solve the multi-class problem, which is expected to be sub-optimal.

Random input attributes that follow a Gaussian distribution have been considered in the context of GP regression in Girard et al. (2003). That work addresses the problem of learning in such a setting by using a Gaussian approximation that matches the mean and the variance of the GP predictive distribution when the input attributes are noisy. The required computations are tractable for some particular covariance functions such as the squared exponential. Such an approach has also been applied in the context of solving control problems using GPs (Deisenroth and Rasmussen, 2011). In that setting, however, the likelihood is a reinforcement learning cost function that has been chosen carefully so that all computations remain tractable. Moment matching to account for random inputs has also been used the context of deep Gaussian processes in which the input to a hidden layer is the output of the previous layer, which is random (Bui et al., 2016). An alternative approximation based on Kalman filters that is potentially more accurate than the one obtained by moment matching has also been considered in the literature in the context of sequential state estimation (Ko et al., 2007) and in the context of GPs with arbitrary non-linear likelihood functions (Steinberg and Bonilla, 2014). Nevertheless, all these works differ from NIMGP in that no inference is been made about the noiseless inputs. More precisely, the mean of the random input attributes is assumed to be known, unlike in our setting, in which it is a latent variable. Moreover, no multi-class classification problems have been specifically addressed.

Solving regression problems in the context of noisy input observations has been addressed by Dallaire et al. (2009) using the moment-matching technique described in the previous paragraph. Accounting for noise in the inputs significantly improves the prediction results of a standard GP. In that work, however, the posterior distribution for the noiseless inputs is computed without conditioning to the observed targets. Note, however, that the likelihood factors $p(y_i|f_i)$ constrain the potential values that the noiseless inputs can take. This is information is ignored when such an approach is used. By contrast, NIMGP is able to consider the label information by learning a specific posterior approximation for each noiseless input, as described in Eq. (8). This posterior approximation is inferred at the same time as the other latent variables of the model. The consequence is that the approach suggested by Dallaire et al. (2009) is expected to be suboptimal. Furthermore, the variance associated to the inputs is assumed to be given. We show that this parameter can be inferred from the observed data in NIMGP and NIMGP$_{\text{NN}}$. Moreover, no classification problems are addressed by those authors.

Another work that has addressed noisy inputs only for test data in the context of GPs is Oakley and O'Hagan (2002). Given the distribution of a test input, the resulting predictive distribution is approximated using Monte Carlo methods. More precisely, samples from the posterior process are generated approximately and the quantities of interest (*e.g.*, the

expected value at the test point) are approximated by sampling from the distribution of the test inputs. The advantage of such a method is that it can take into account arbitrary distributions for the noisy inputs. The limitation is, however, that only regression problems are addressed, the training inputs are assumed to be noiseless, and the distribution of the noisy test inputs is assumed to be known beforehand.

Noisy inputs have also been considered in the context of distribution regression in which the inputs of the regression problem are data sets and/or probability distributions (Law et al., 2018). In this case, a kernel mean embedding can be used a *summary* or feature representation of each data set (Muandet et al., 2017). This feature representation can be estimated from empirical data, and the variance of the estimation is expected to be reduced with the size of each data set. In the referred work it is described how to account for uncertainty in the estimation of the kernel mean embedding in the context of regression problems using a Bayesian approach. A sophisticated method based on Hamilton Monte Carlo (HMC) is suggested to approximate the required computations. As described previously, this work only considers regression problems, assumes that the distribution of the inputs is known before hand and, moreover, it could be limited by the expensive cost of running a Markov Chain using HMC.

Noisy inputs have also been consider in the context of other methods not directly related to GPs. More precisely, other approaches to deal with input noise involve using robust features in the context of multi-class SVMs (Rabaoui et al., 2008) or enhancements of fuzzy models (Ge and Wang, 2007). The first work can be understood as a pre-processing step in which robust features are generated and used for training with the goal of reducing the effect of background noise in a sound recognition system. These robust features are, however, specific of the application domain conspired. Namely, sounds recognition. They are not expected to be useful in other classification problems. The second work focuses exclusively on linear regression models and hence cannot address multi-class classification problems, as the ones considered in our work.

5. Experiments

In this section we carry out several experiments to evaluate the performance of the proposed method for multi-class GP classification with input noise. More precisely, we compare the performance of a standard multi-class Gaussian process that does not consider noise in the inputs (MGP) and the three proposed methods. Namely, the approach described in Section 3.1 (NIMGP), the variant described in Section 3.2 where the parameters of the Gaussian posterior approximation are computed using a neural network (NIMGP$_{\text{NN}}$) and the method proposed in Section 3.3 (NIMGP$_{\text{FO}}$), which is based on the work of Mchutchon and Rasmussen (2011), and uses a first order approximation to account for input noise. The experiments considered include both in synthetic and real data. All the experiments carried out (except those related to the MNIST data set) involve 100 repetitions and we report average results. These are detailed in the following sections.

All the methods described have been implemented in Tensorflow (Abadi et al., 2015). The source code to reproduce all the experiments carried out is available online at `https://github.com/cvillacampa/GPInputNoise`. In these experiments, for each GP, we have employed a squared exponential covariance function with automatic relevance determina-

tion (Rasmussen and Williams, 2006). All hyper-parameters, including the GP amplitude parameter, the length-scales and the level of additive Gaussian noise have been tuned by maximizing the ELBO. The class noise level has been set equal to 10^{-3}, and kept fixed during training as in Hensman et al. (2015b). Unless indicated differently, we have set the number of inducing points for the sparse Gaussian process to the minimum of 100 and 5% of the total number of points. For the optimization of the ELBO we have used the ADAM optimizer with learning rate equal to 0.01, the number of epochs has been set to 750 and the mini-batch size to 50 (Kingma and Ba, 2015). This number of epochs seems to guarantee the convergence of the optimization process. All other ADAM parameters have been set equal to their default value. In NIMGP$_{NN}$ the neural network has 50 hidden units and one hidden layer. The activation function is set to be ReLu. Finally, the number of Monte Carlo samples used to approximate the predictive distribution in NIMGP and NIMGP$_{NN}$ is set to 300.

5.1 Illustrative Toy Problem

Before performing a fairly complete study on more realistic examples, we show here the results of the three proposed methods on a one-dimensional synthetic data set with three classes. This data set is simple so that it can be analyzed in detail and the optimal predictive distribution provided by the Bayes classifier can be computed in closed-form. The data set is generated by sampling the latent functions from the GP prior using specific hyper-parameter values and then applying the labeling rule in (1). The input locations have been generated randomly by drawing from a uniform distribution in the $[-3, 3]$ interval. Then, we add a Gaussian noise to each observation $\mathbf{x_i}$ to generate $\tilde{\mathbf{x}}_i$, with standard deviation $\mathbf{V}_i = 0.1\mathbf{I}$ for $i = 1, \ldots, N$. We consider $1,000$ training instances and the number of inducing points $M = 100$. The mini-batch size is set to 200 points. Since in this experiment the variance of the input noise is known beforehand we do not infer its value from the observed data and rather specify its actual value in each method. The number of test points is $1,000$. In NIMGP$_{NN}$ the neural network is set to have 2 hidden layers with 50 units each.

We have trained each method on this data set and compared the resulting predictive distribution with that of the optimal Bayes classifier, which we can compute in closed form since we know the generating process of the labels. Figure 2 shows, for each method, as a function of the input value x, the predicted distribution for each of the three classes[1]. The observed labels for each data point are shown at the top of each figure as small vertical bars in green, blue and red colors, depending on the class label. We observe that each method produces decision boundaries that agree with the optimal ones (*i.e.*, all methods predict the class label that has the largest probability according to the optimal Bayes classifier). However, the predictive distributions produced differ from one method to another. More precisely, the first method, *i.e.*, MGP (top-left), which ignores the input noise, does produce a predictive distribution that is significantly different from the optimal one, especially in regions of the input space that are close to the decision boundaries. This method produces a predictive distribution that is too confident. The closest predictive distribution to the op-

---

1. The wiggles in the prediction probability for some of the classes for the NIMGP and NIMGP$_{NN}$ are produced by the Monte Carlo approximation of the predictive distribution. Here we use a Monte Carlo average across 700 samples.
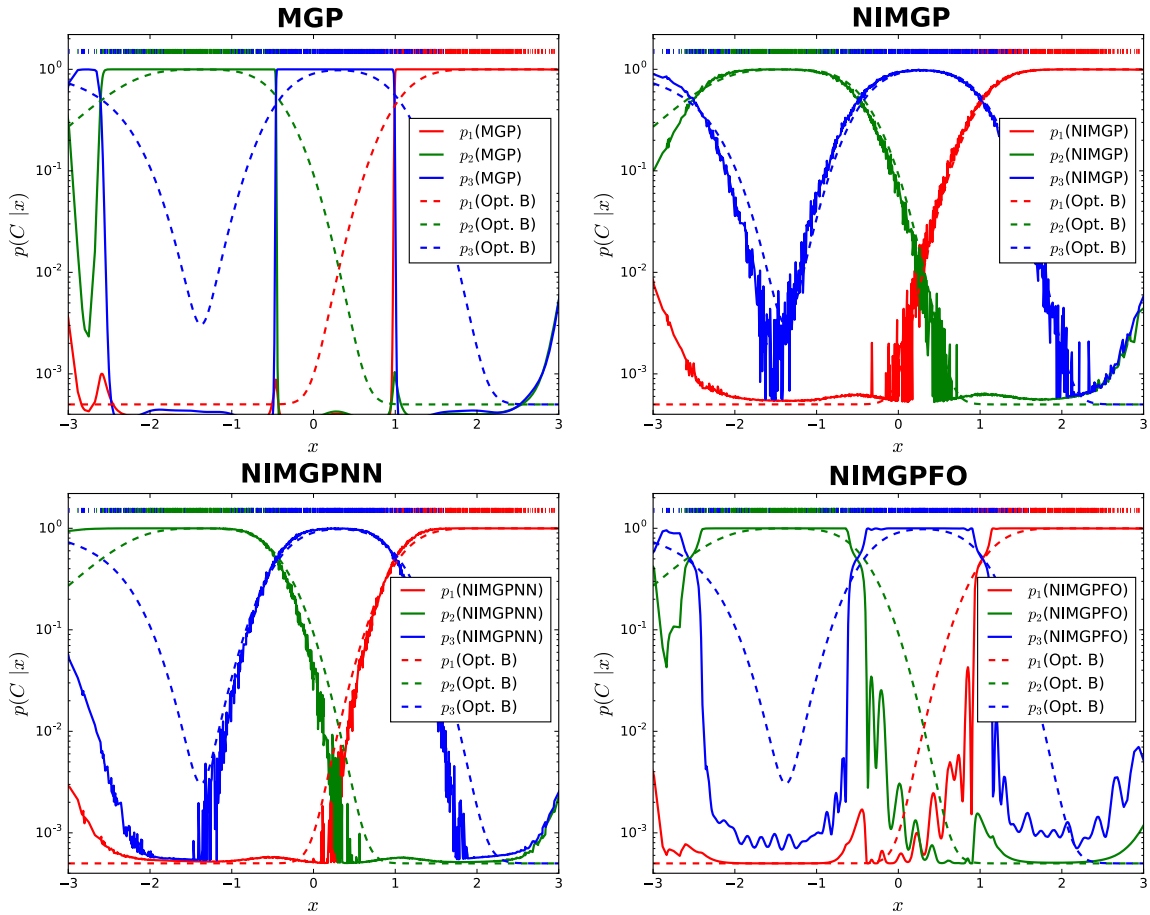
Figure 2: Comparison of the predictive distribution computed by each method and the optimal Bayes prediction (shown as a dashed line). The observed labels are shown as small vertical bars at the top of the plot in green, blue and red. (Top-left) Model without input noise (MGP), (top-right) NIMGP model, (bottom-left) NIMGP$_{NN}$ and (bottom-right) NIMGP$_{FO}$. While the decision boundaries of each method agree with the optimal ones, NIMGP is the method that more accurately computes the predictive probabilities of each class label. By contrast MGP produces predictions that are too confident.

timal one is obtained by NIMGP (top-right), followed by NIMGP$_{NN}$ (bottom-left). Finally, NIMGP$_{FO}$ (bottom-right) seems to improve the results of MGP, but is far from NIMGP.

Figure 2 shows very clearly the advantage of including the input noise when estimating the predictive distribution $p(c|x)$ for each class label $c = 1, 2, 3$, given a new test point x. Indeed, the proposed models reproduce more closely the optimal predictive distribution of the Bayes classifier. By contrast, the model that ignores the input noise, *i.e.*, MGP fails to produce an accurate predictive distribution. Therefore, we expect to obtained better results for the proposed methods in terms of the log-likelihood of the test labels.

Table 1 shows the prediction error of each method and the corresponding negative test log-likelihood (NLL). Standard deviations are estimated using the bootstrap. Importantly, NLL can be understood as a measure of the quality of the predictive distribution. The smaller the better. We observe that while the prediction error of each method is similar (since the decision boundaries produced are similar) the NLL of the proposed methods is significantly better than the one provided by MGP, the method that ignores input noise. These results highlight the importance of accurately modeling the input noise to obtain better predictive distributions, which can play a critical role when one is interested in the confidence on the decisions to be made in terms of the classifier output.

	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
Test error	**0.125±0.011**	0.129 ± 0.011	**0.125±0.010**	0.128 ± 0.010
Test NLL	0.884 ± 0.079	**0.286±0.017**	0.347 ± 0.030	0.495 ± 0.046

Table 1: Test error and negative test log-likelihood (NLL) for the one-dimensional toy experiment.

5.2 Synthetic Experiments

Next, we compare the methods on 100 synthetic two-dimensional classification problems with three classes. As in the case of the one-dimensional data set described above, these problems are generated by sampling the latent functions from a GP prior with the squared exponential function and then applying the labeling rule in (1). The GP hyper-parameters employed are $\sigma^2 = 0.5$, $\sigma_0^2 = 0$, $\ell_j = 2, \forall j$. The input vectors $\mathbf{x}_i$ are chosen uniformly in the box $[-2.5, 2.5]^2$. Then, we add three different levels of random noise to each observation $\mathbf{x}_i$, i.e., $\mathbf{V}_i = \{0.1\mathbf{I}, 0.25\mathbf{I}, 0.5\mathbf{I}\}$. The interest of these experiments is to evaluate the proposed methods in a controlled setting in which the expected optimal model to explain the observed data is a multi-class GP classifier and we can control the level of input noise in the data. In the next section we carry out experiments with real data sets. The number of training and test instances, inducing points, mini-batch size and parameters of the ADAM optimizer are the same as in the previous experiment. Again, since the level of injected noise is known in these experiments, we directly codify this information in each method. Figure 3 shows a sample data set before and after the noise injection.

The average results obtained on the 100 data sets, for each method, are displayed in Tables 2 and 3. We report results when the variance of the input noise is given beforehand (given), and when it is learned from the observed data (learned). In this last case, the lower bound given by Eq. (10) is maximized to estimate the input noise variance, as described in Section 3.4. We assume the same variance of the Gaussian input noise for each attribute.

In the first scenario (given), we observe that, in terms of the negative test log-likelihood, all the proposed methods improve over MGP, i.e., the standard GP multi-class classifier that ignores input noise. Among the proposed methods, the best performing one is NIMGP, closely followed by NIMGP$_{NN}$. Therefore, these experiments highlight the benefits of using a neural network to compute the parameters of the posterior approximation $q(\mathbf{x}_i)$. Specifically, there is no performance degradation. The method that is based on the first order
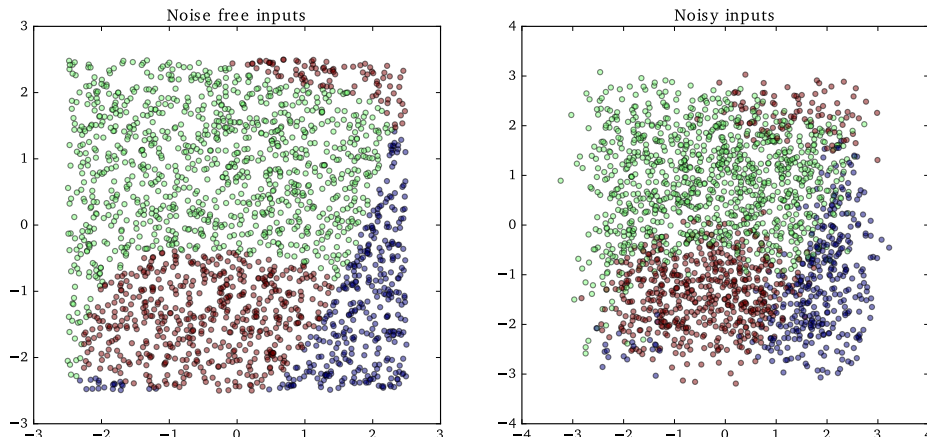
Figure 3: Sample synthetic classification problem with three class labels. (left) Input data and associated class label before the noise injection in the observed attributes. (right) Input data and associated class labels after the noise injection. The variance of the Gaussian noise is set equal to 0.1. Best seen in color.

approximation, *i.e.*, NIMGP$_\text{FO}$, also improves over MGP, but the differences are smaller. In terms of the prediction error the differences are much smaller. However, in spite of this, the proposed methods improve the results of MGP. Among them, again NIMGP is the best method, except when $\mathbf{V}_i = 0.5\mathbf{I}$. In that case, NIMGP$_\text{FO}$ performs best, but the differences are very small. Summing up, the results obtained agree with the ones obtained in the one-dimensional problem. Namely, the proposed approaches significantly improve the quality of the predictive distribution in terms of the neg. test log-likelihood. The prediction error is, however, similar.

In the second scenario, *i.e.*, when the variance of the input noise is learned from the data, we observe that NIMGP$_\text{FO}$ performs more or less similarly to MGP, and that NIMGP and NIMGP$_\text{NN}$ obtain better results in terms of the test log-likelihood, particularly in the case of the second method. In terms of the prediction error all the methods give similar results, as in the previous scenario. The reason for these results is that NIMGP$_\text{NN}$ seems to be the only method that can provide sensible estimates of the input noise variance from the observed data. This is confirmed by Figure 4, which shows a histogram of the input noise variances inferred by each method across the 100 repetitions of the experiments. Note that NIMGP and NIMGP$_\text{FO}$ seem to always estimate a variance level that is close to zero. By contrast, NIMGP$_\text{NN}$ estimates an input noise variance that is very similar to the actual one. This is an unexpected observation. We believe that the better results obtained by NIMGP$_\text{NN}$ over NIMGP can be a consequence of the regularization properties introduced by the neural network employed in this method. This is confirmed by the experiments carried out in the next section. The bad results of NIMGP$_\text{FO}$ could be related to the linear approximations introduced in this method.

In Appendix B we report extra experimental results on the synthetic data sets described using other configurations (*i.e.*, varying the number of classes, the number of dimensions and

	Noise	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
Given	0.1	0.758±0.022	**0.0256±0.007**	0.0265±0.008	0.321±0.009
	0.25	1.14±0.033	**0.0369±0.011**	0.0388±0.012	0.53±0.015
	0.5	1.537±0.041	**0.0493±0.012**	0.0526±0.014	0.77±0.021
Learned	0.1	0.758±0.022	0.61±0.016	**0.28±0.0087**	0.72±0.021
	0.25	1.14±0.033	0.87±0.021	**0.4±0.012**	1.1±0.032
	0.5	1.537±0.04	0.93±0.017	**0.53±0.014**	1.5±0.041

Table 2: Average Neg. Test Log Likelihood for each method on the synthetic problems when the variance of the input noise is given and learned from the data.

	Noise	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
Given	0.1	0.113±0.003	**0.108±0.003**	0.108±0.003	0.109±0.003
	0.25	0.164±0.006	**0.158±0.005**	0.158±0.005	0.158±0.005
	0.5	0.218±0.006	0.21±0.058	0.22±0.063	**0.21±0.006**
Learned	0.1	0.113±0.003	**0.11±0.003**	**0.11±0.003**	0.11±0.003
	0.25	0.164±0.006	0.17±0.005	0.17±0.005	**0.16±0.005**
	0.5	0.218±0.006	0.23±0.006	**0.21±0.006**	0.22±0.006

Table 3: Average Test Error for each method on the synthetic problems when the variance of the input noise is given and learned from the data.

the number of training instances). We also report results for two other baselines. These are (i) using MGP and augmenting the training set with samples from the posterior distribution of each data instance assuming a uniform prior; and (ii) using MGP but sampling the input attributes for each mini-batch from the previous distribution. The results obtained for the other data configurations are similar to the ones reported here. Moreover, the other baselines do not seem to improve the results of MGP at all, or the improvements are significantly smaller the ones obtained by NIMGP, NIMGP$_{NN}$ and NIMGP$_{FO}$.

5.3 Experiments on data sets Extracted from the UCI Repository

Another set of experiments evaluates the proposed methods on 8 different multi-class data sets extracted from the UCI repository (Dua and Graff, 2017). Table 4 displays the characteristics of the data sets considered. For each of these data sets, we consider 100 splits into train and test, containing 90% and 10% of the data respectively. Unlike in the previous experiments, in these data sets, a GP multi-class classifier need not be optimal. Furthermore, the input attributes may already be contaminated with additive noise. To assess the benefits of considering such noise during the learning process, we have considered four different setups. In each one, we inject Gaussian noise in the observed attributes with different variances. Namely, 0.0, 0.1, 0.25 and 0.5. This will allow to evaluate the different methods in a setting in which there may be or may be not input noise due to the particular characteristics of the problem addressed (*i.e.* when the variance of the injected noise is equal to 0.0), and also for increasing levels of input noise (*i.e.*, when the variance of the injected noise is equal
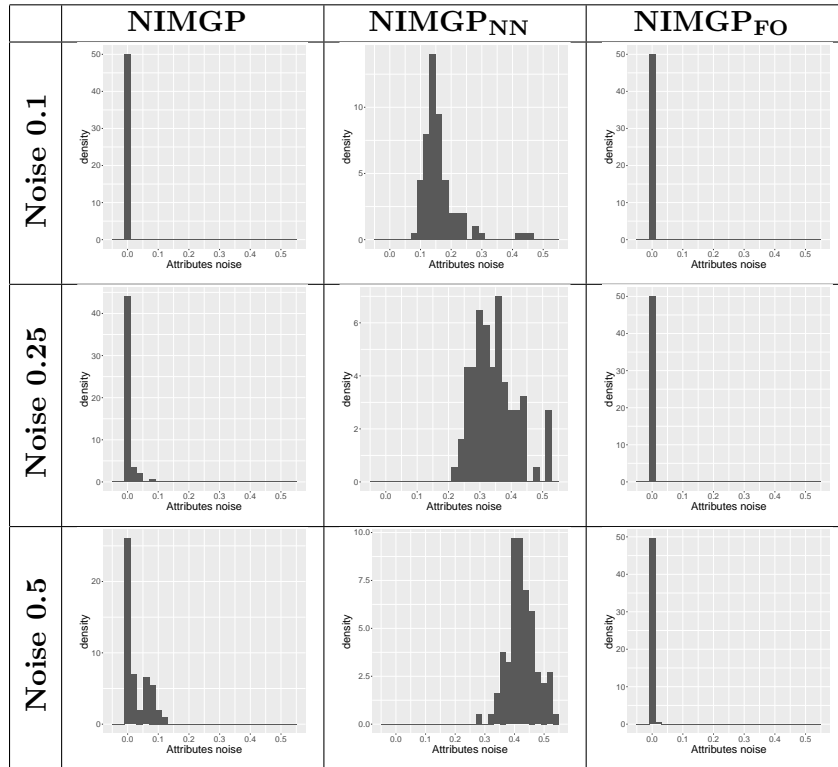
Figure 4: Histogram of the estimated values (across the 100 experiments) for the variance of the input noise, for each method. Note that NIMGP$_{NN}$ is the only method producing sensible estimates of the input noise variance in each case. The other methods most of the times estimate values for the noise variance that are almost zero.

to 0.1, 0.25 and 0.5). This noise injection process is done after a standardization step in which the observed attributes are normalized to have zero mean and unit variance in the training set. All methods are trained for $1,000$ epochs using a mini-batch size of 50. In these experiments, in each of the proposed methods, the level of noise is learned during the training process by maximizing the ELBO. The reason for this is that the actual level of noise in the input attributes need not be equal to the injected level of noise.

Table 5 shows the average results obtained for each method in terms of the negative test log-likelihood, for each level of noise considered. We also report the mean rank for each method across data sets and splits. If a method is always the best one, it will receive a mean rank equal to 1. Conversely, if a method is always worst, it will receive a mean rank equal to 4. Therefore, in general lower is better. We can see that on average, the proposed methods improve over MGP and the method that works the best (according to the mean rank) is NIMGP$_{NN}$, even for the case where we do not introduce noise in the inputs. This suggests that these data sets have already some noise in the inputs. Also, as we increase the noise level the mean rank for MGP and NIMGP is worsen and NIMGP$_{NN}$ and NIMGP$_{FO}$

both improve. The fact that NIMGP$_{\text{NN}}$ and NIMGP$_{\text{FO}}$ give better results than NIMGP as we increase the level of noise indicates that the use of the neural network and the first order approximation act as a regularizer with better generalization properties (Shu et al., 2018).

Table 6 shows the average results obtained for each method in terms of the prediction error. In this case, we do not observe big differences among the different methods. Moreover, the methods that take into account the noise in the inputs do not improve the prediction error as we increase the noise level. These small differences in terms of the error can be due to each method having similar decision boundaries, even when we obtain better predictive distributions in terms of the test log-likelihood, as illustrated in Section 5.1.

5.4 Experiments on the MNIST data set

In this section we consider a data set in which sparse GPs are needed in order to train a multi-class classifier based on GPs. Namely, the MNIST data set (LeCun et al., 1998). This data set has 10 different class labels and $60,000$ training instances lying in a 28×28 dimensional space. The test set has $10,000$ data instances. We consider a similar setup to the previous experiments and inject noise in the inputs (after a standardization to guarantee zero mean and unit standard deviation on the input attributes) with variances equal to 0.0, 0.1, 0.25 and 0.5. The level of noise is also learned during the training process by maximizing the ELBO. The mini-batch size is set to 200 and the number of training epochs is set to 350. This number of epochs seems to be large enough to guarantee the convergence of the different methods evaluated. In the case of NIMGP$_{\text{NN}}$ the neural network has 250 units and two hidden layers. We also slightly modified the neural network so that at the beginning of the training process it outputs as the mean the noisy observed attributes $\tilde{\mathbf{x}}_i$ fed at the input. The number of Monte Carlo samples used to approximate the predictive distribution in NIMGP and NIMGP$_{\text{NN}}$ is set to 500. We use a bigger number of Monte Carlo samples in these experiments because of the bigger size of the classification problem (10 classes) and the input dimensionality, $i.e.$, 784 dimensions. All the computations are sped-up by using a TESLA P100 GPU for training. In these experiments we use a polynomial kernel with automatic relevance determination since there is evidence supporting that it works better on this data set (Henao and Winther, 2012).

data set	#Instances	#Attributes	#Classes
Glass	214	9	6
New-thyroid	215	5	3
Satellite	6435	36	6
Svmguide2	391	20	3
Vehicle	846	18	4
Vowel	540	10	6
Waveform	1000	21	3
Wine	178	13	3

Table 4: Characteristics of the data sets extracted from the UCI repository.

		MGP	NIMGP	NIMGP$_{\text{NN}}$	NIMGP$_{\text{FO}}$
Noise = 0.0	glass	1.63±0.048	1.28±0.04	**1.17 ±0.033**	1.19±0.033
	new-thyroid	0.096±0.007	**0.083 ±0.006**	0.122±0.006	0.113±0.005
	satellite	0.5±0.007	0.363±0.005	**0.281 ±0.002**	0.316±0.003
	svmguide2	0.594±0.024	0.586±0.023	**0.519 ±0.018**	0.531±0.02
	vehicle	0.638±0.019	0.514±0.019	**0.408 ±0.005**	0.497±0.006
	vowel	0.415±0.025	**0.278 ±0.019**	0.321±0.012	0.445±0.015
	waveform	0.676±0.017	0.657±0.015	**0.335 ±0.006**	0.451±0.01
	wine	**0.054 ±0.004**	0.056±0.004	0.074±0.004	0.065±0.004
	Mean rank	3.05±0.0429	2.39±0.0377	**2.05±0.0486**	2.51±0.0295
Noise = 0.1	glass	1.71±0.051	1.91±0.055	**1.33 ±0.034**	1.37±0.041
	new-thyroid	0.278±0.025	0.303±0.026	**0.19 ±0.011**	0.201±0.013
	satellite	0.703±0.008	0.613±0.007	**0.347 ±0.003**	0.421±0.004
	svmguide2	0.663±0.025	0.655±0.024	**0.565 ±0.018**	0.596±0.021
	vehicle	1.25±0.027	1.28±0.026	**0.612 ±0.007**	0.795±0.014
	vowel	0.836±0.026	0.815±0.025	**0.656 ±0.012**	0.673±0.018
	waveform	0.752±0.017	0.734±0.016	**0.381 ±0.006**	0.52±0.011
	wine	**0.087 ±0.007**	0.089±0.007	0.097±0.006	0.093±0.007
	Mean rank	3.19±0.038	3.06±0.03	**1.71±0.037**	2.04±0.022
Noise = 0.25	glass	1.89±0.053	1.96±0.06	**1.36 ±0.034**	1.45±0.037
	new-thyroid	0.445±0.035	0.472±0.034	**0.271 ±0.015**	0.302±0.02
	satellite	0.835±0.009	0.77±0.008	**0.409 ±0.003**	0.472±0.004
	svmguide2	0.761±0.025	0.756±0.025	**0.627 ±0.015**	0.638±0.019
	vehicle	1.61±0.031	1.65±0.03	**0.783 ±0.008**	0.967±0.017
	vowel	1.37±0.037	1.38±0.033	1.04±0.013	**0.943 ±0.017**
	waveform	0.849±0.018	0.836±0.018	**0.434 ±0.006**	0.519±0.009
	wine	**0.134 ±0.011**	0.136±0.011	0.15±0.008	0.141±0.008
	Mean rank	3.25±0.033	3.16±0.028	**1.68±0.037**	1.92±0.025
Noise = 0.5	glass	2.03±0.053	2.01±0.051	**1.45 ±0.034**	1.52±0.038
	new-thyroid	0.565±0.038	0.623±0.04	**0.369 ±0.018**	0.381±0.021
	satellite	0.973±0.009	0.932±0.009	**0.491 ±0.003**	0.531±0.004
	svmguide2	0.877±0.025	0.878±0.025	0.706±0.013	**0.702 ±0.018**
	vehicle	1.93±0.032	1.99±0.031	**0.994 ±0.006**	1.1±0.016
	vowel	1.99±0.038	2.08±0.037	1.33±0.012	**1.25 ±0.018**
	waveform	1.01±0.021	0.984±0.021	**0.503 ±0.006**	0.565±0.01
	wine	0.253±0.017	0.264±0.017	0.24±0.01	**0.236 ±0.011**
	Mean rank	3.33±0.027	3.25±0.028	**1.63±0.03**	1.79±0.027

Table 5: Average neg. test log likelihood for the experiments on the UCI data sets.

		MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
Noise = 0.0	glass	0.346±0.008	0.387±0.01	0.375±0.009	**0.345 ±0.009**
	new-thyroid	0.041±0.004	**0.031 ±0.004**	0.042±0.005	0.044±0.004
	satellite	0.092±0.001	**0.092 ±0.001**	0.118±0.001	0.093±0.001
	svmguide2	0.166±0.006	**0.165 ±0.006**	0.174±0.006	**0.165 ±0.006**
	vehicle	0.16±0.004	**0.155 ±0.004**	0.216±0.004	0.159±0.004
	vowel	0.07±0.004	**0.054 ±0.003**	0.096±0.004	0.068±0.004
	waveform	0.155±0.003	0.153±0.003	**0.14 ±0.003**	0.155±0.003
	wine	0.024±0.003	0.024±0.003	**0.019 ±0.003**	0.022±0.003
	Mean rank	2.37±0.0349	2.32±0.0393	2.92±0.0461	2.39±0.0326
Noise = 0.1	glass	0.389±0.009	0.412±0.01	0.413±0.009	**0.387 ±0.009**
	new-thyroid	0.076±0.006	0.078±0.006	**0.07 ±0.005**	0.075±0.006
	satellite	0.128±0.001	**0.127 ±0.001**	0.142±0.001	0.128±0.001
	svmguide2	0.198±0.006	0.198±0.006	**0.192 ±0.006**	0.198±0.006
	vehicle	0.291±0.005	0.295±0.005	**0.291 ±0.005**	0.291±0.005
	vowel	0.217±0.005	**0.217 ±0.004**	0.246±0.005	0.218±0.005
	waveform	0.162±0.003	0.163±0.003	**0.154 ±0.003**	0.162±0.003
	wine	0.034±0.004	0.036±0.004	**0.033 ±0.004**	0.033±0.004
	Mean rank	2.44±0.036	2.51±0.04	2.64±0.0503	2.41±0.0357
Noise = 0.25	glass	0.433±0.009	0.456±0.01	0.467±0.01	**0.432 ±0.009**
	new-thyroid	0.101±0.007	0.106±0.006	**0.093 ±0.006**	0.098±0.007
	satellite	0.155±0.001	**0.155 ±0.001**	0.163±0.001	0.155±0.001
	svmguide2	0.217±0.006	0.218±0.006	0.237±0.006	**0.216 ±0.006**
	vehicle	0.357±0.006	0.366±0.006	**0.351 ±0.006**	0.36±0.006
	vowel	0.351±0.007	**0.345 ±0.006**	0.432±0.008	0.349±0.007
	waveform	0.193±0.003	0.193±0.004	**0.188 ±0.003**	0.193±0.003
	wine	0.052±0.006	0.055±0.006	**0.049 ±0.005**	0.052±0.005
	Mean rank	2.39±0.0316	2.48±0.0377	2.71±0.0444	2.42±0.0356
Noise = 0.5	glass	**0.47 ±0.009**	0.501±0.01	0.51±0.011	0.473±0.01
	new-thyroid	0.125±0.007	0.148±0.008	0.139±0.007	**0.122 ±0.007**
	satellite	0.181±0.002	**0.181 ±0.002**	0.188±0.002	0.181±0.001
	svmguide2	0.256±0.007	**0.256 ±0.007**	0.288±0.006	**0.256 ±0.007**
	vehicle	0.428±0.006	**0.422 ±0.006**	0.445±0.005	0.424±0.006
	vowel	**0.473 ±0.007**	0.478±0.008	0.565±0.007	0.475±0.007
	waveform	0.225±0.004	0.225±0.004	**0.222 ±0.004**	0.225±0.004
	wine	0.092±0.006	0.092±0.006	0.093±0.006	**0.088 ±0.007**
	Mean rank	2.35±0.0354	2.46±0.0396	2.91±0.0389	2.29±0.0353

Table 6: Average Test Error for experiments on UCI data sets

Noise	Metric	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
0.0	NLL	0.0649	0.0652	**0.0630**	0.0651
	Error	0.0203	0.02	**0.0196**	0.0202
0.1	NLL	0.0718	0.0712	**0.0704**	0.0720
	Error	**0.0215**	0.0221	**0.0215**	0.0216
0.25	NLL	0.0878	0.0879	**0.0852**	0.0880
	Error	0.0265	0.0275	**0.0253**	0.0265
0.5	NLL	0.1023	0.1024	**0.1005**	0.1024
	Error	0.029	0.0293	**0.0284**	0.0288

Table 7: Average test error and neg. test log-likelihood (NLL) of each method on the MNIST data set.

The results obtained in these experiments, for each method, are displayed in Table 7. The results obtained are similar to others reported in the literature (Henao and Winther, 2012; Hensman et al., 2015b; Villacampa-Calvo and Hernández-Lobato, 2017). We observe that the proposed methods always outperform the MGP, *i.e.*, the multi-class GP that ignores the input noise. Both in terms of the test error and the negative test log-likelihood. In this case, however, the gains are small. We believe this is because this data set is particularly challenging for GPs (Van der Wilk et al., 2017).

Figure 5 shows a histogram of the estimated variances for the input noise of each of the 784 attributes of the MNIST data set, for each method and level of injected noise. We observe that in general all the methods except NIMGP$_{NN}$ tend to estimate input variances that are very similar for each attribute and that are close to zero. By contrast, NIMGP$_{NN}$ estimates different values for the input noise variance associated to each attribute. Furthermore, these values are significantly bigger than the ones estimated by the other methods. This may explain the better results obtained by this method on this data set. However, in spite of this, the estimated variances are much smaller than the level of injected noise. This means that the method is underestimating the level of input noise. We believe that this can be a consequence of the difficulty of this data set for models based on GPs, as commented by Van der Wilk et al. (2017).

Table 8 shows the average training time employed on each epoch for each method. We observe that MGP, NIMGP and NIMGP$_{NN}$ have similar training times. However, NIMGP$_{FO}$ takes a significantly larger amount of training time on each epoch. This is due to the extra cost of computing the gradients of the GP predictive mean, for each latent function. This is an expensive operation. Recall that the input dimensionality of this data set is high (784 dimensions) and also the number of class labels (10 class labels).

	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
Avg. Time	7.19±0.007	11.93±0.011	7.62±0.011	24.01±0.074

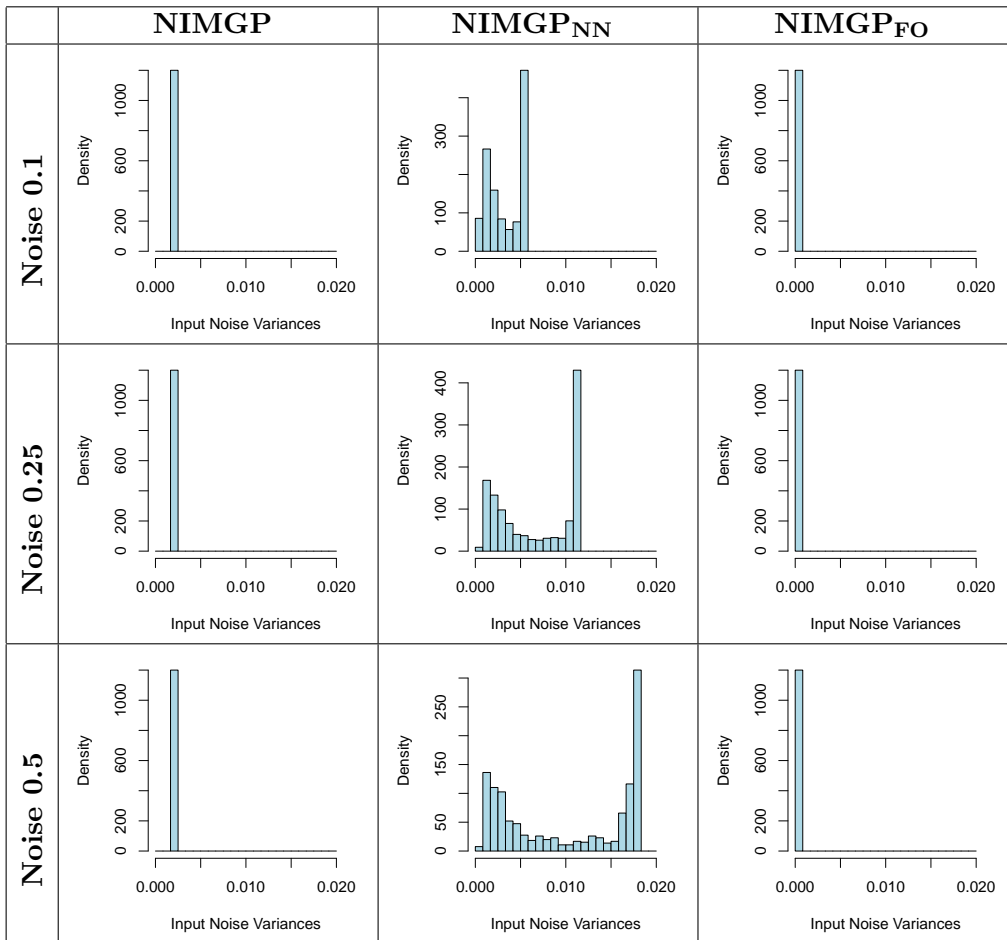Table 8: Average time per epoch, in seconds, for each method on the MNIST data set.

Figure 5: Histogram of the estimated values for the variance of the input noise, for each method, across the 784 dimensions of the MNIST data set. Note that NIMGP$_{\text{NN}}$ is the only method producing sensible estimates of the input noise variance in each case. The other methods most of the times estimate values for the noise variance that are almost zero.

5.5 Active Learning Experiments

We have observed that modeling input noise can lead to better prediction results in terms of the test log-likelihood while the prediction error is similar in most of the cases. Therefore, it seems that the main benefit is a more accurate predictive distribution. In this section, we consider an active learning experiment on the *Waveform* data set to illustrate that a better predictive distribution can be useful to improve the generalization error. In active learning the task of interest is to identify which examples should be labeled and introduced into the training set to produce a better prediction model. We aim at showing that the predictive distribution can be used to identify the training examples that are expected to be most useful for learning a particular task (Settles, 2009). In particular, a popular choice is to

consider the examples whose associated labels the model is more uncertain about (Seeger, 2008). Therefore, a better predictive distribution should be translated into better active learning results. That is, the predictive performance should improve more as more and more examples are introduced into the training set, using the aforementioned strategy.

With the goal described, we begin with an initial training set, a test set to evaluate the performance, and a validation set from which we will iteratively pick up a new point to be labeled and included the training set. At each iteration, we will select the point for which the predictive entropy is highest. In order to validate such an approach, we compare this selection criterium with an approach that selects the next point at random from the validation set. We randomly split the Waveform data set into 100 points for the initial training set, 500 points for testing and 400 points for validation. We add 100 new points to the training set using the active learning strategies described. We report results for each method. Namely, MGP, NIMGP, NIMGP$_{NN}$ and NIMGP$_{FO}$ and each level of noise considered in the experiments of Section 5.3. All methods are trained using ADAM with a learning rate of 10^{-3} the first time for $1,000$ epochs, and then re-trained each time a new point is added to the training set for another $1,000$ epochs, reusing the solution that we have obtained in the previous iteration. The number of inducing points is set to 50. We report averages over 100 repetitions of the experiments using different splits of the data. In these experiments we also infer the variance of the input noise from the observed data.

Figure 6 shows the test error as a function of the number of new added points, for each method and each input noise variance injected in the data. In the top row, the new points have been added by using the active learning approach described. That is, the point whose associated predictive entropy is highest, according to the predictive distribution of the corresponding model. In the bottom row, the new points have been added at random from the validation set. We observe that the test error is always lower when the active learning selection approach based on the entropy is used. This illustrates that a good predictive distribution can lead to better generalización error. We also observe that the prediction error increases at the beginning as the number of new training points considered increases. Eventually it plateaus and begins to decrease. We believe that this behavior could be related to over-fitting happening as a consequence of the procedure to estimate the model hyper-parameters, which maximizes an estimate of the marginal likelihood in all the methods. If the number of training points is fairly small, this approach can produce over-fitting (Rasmussen and Williams, 2006). As in the experiments carried out in Section 5.3, the best performing method is NIMGP$_{NN}$.

Figure 6 shows that an active learning strategy can be useful to produce better prediction results. We now analyze in which method such a strategy leads to bigger improvements. For this, Figure 7 shows the test error reduction w.r.t. the initial test error for each method and each values of the input noise variance. We observe that the reduction is higher for NIMGP$_{NN}$ than for the other methods, and the difference is bigger between the methods as we increase the variance of the input noise. This suggests that the predictive distribution obtained by NIMGP$_{NN}$ is better. Moreover, it illustrates the potential benefits of considering in the classifier that the observed attributes have been contaminated with additive Gaussian noise.
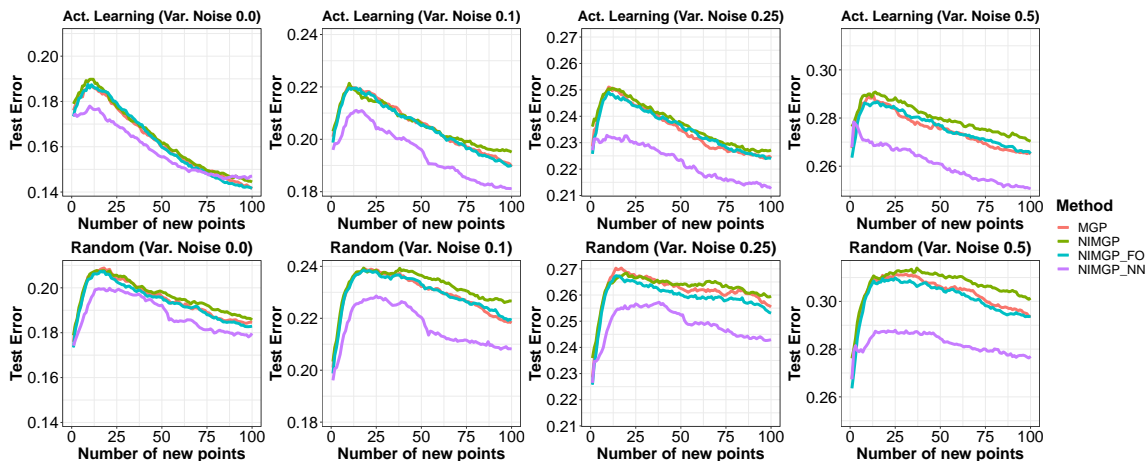
Figure 6: Test error on the Waveform data set for different values of the input noise variance (0.0, 0.1, 0.25 and 0.5) as a function of the number of added points to the training set, selected using an active learning approach (top) and or at random (bottom).

5.6 Experiments on a data set Coming from Astrophysics

In this last experimental section we describe the results obtained by each method on a three class data set coming from the astrophysics domain. Importantly, in this data the errors in some of the observed inputs are available at training time. Therefore, it is suited to be analyzed using the methods proposed in our paper. As briefly commented in the introduction, the data set consists of a series of attributes measured for a set of point-like astrophysical sources located all over the sky which have already been identified (distinguished from the diffuse background of photon emission) by the Fermi-LAT collaboration. Such catalogue of sources is fully public and can be downloaded from Collaboration (2019), while a detailed description can be found in Abdollahi et al. (2020).

Among the available attributes of the sources, there is the position in the sky, the flux of photons in different energy bins, the significance of detection (according to a specified test statistics), the variability of the flux over some period of time, characteristics of the energy spectrum, and many others. Among these, we have only taken into account the following attributes: 1) the photon flux between 1 GeV and 100 GeV, 2) the detection significance (in number of sigmas), 3) the curvature significance [2], 4) the pivot energy [3], 5) the index α [4] and 6) the index β [5]. Of these attributes, the flux, the index α and the index β come with associated estimated error bars, whereas the rest, by definition, do not. The choice of these attributes among all the available ones is motivated from a physics point of view, where it has been shown (see Abdollahi et al., 2020) that some combinations of them

---

[2]. Defined as the significance, in number of sigmas, of the fit improvement between PowerLaw and Log-Parabola fits of the energy spectrum.

[3]. Defined as the energy at which the error in the differential photon flux is minimum.

[4]. Defined as the spectral slope at pivot energy of a log-parabola fit of the spectrum.

[5]. Defined as the *curvature* parameter of a log-parabola fit of the spectrum.
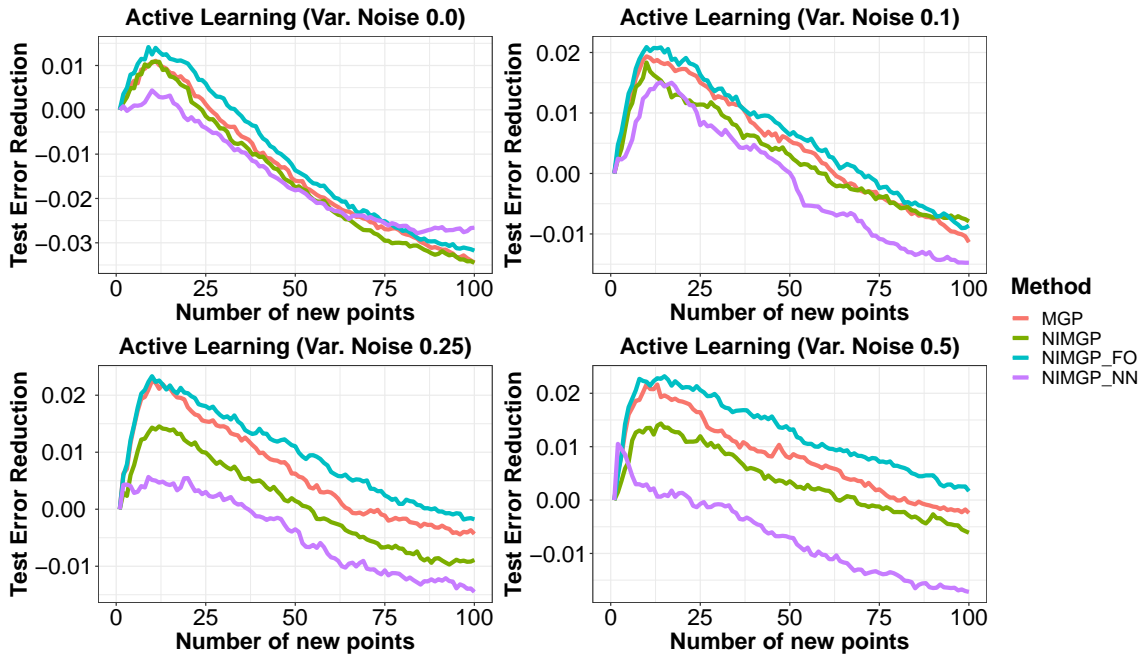
Figure 7: Test error reduction on the Waveform data set as a function of the number of new added points to the training set, selected using an active learning approach.

offer a promising discrimination power among the classes of point-like sources considered. The reader may wonder about the motivation of two unusual attributes: the detection significance and the curvature significance, being both goodness-of-fit quantities. While the detection significance is indeed partially correlated with the photon flux (which is one of the attributes), the curvature significance a priori may contribute to the discrimination power, since different classes have relatively different spectral shapes. We have checked that the performance of the model without these two attributes is somewhat smaller (see Appendix B.4), although compatible at the three standard deviation level, with the performance obtained with the full list of attributes.

On the other hand, the catalogue offers as well several classes of sources. Here, for simplicity, we have chosen three classes: pulsars (in the catalogue, `psr`), blazars (`bll`) and quasars (`fsrq`) [6], which are also the most abundant classes among all the catalogue. The final data set, after pre-processing and pre-selection[7] contain 454 sources (points), of which 184 are pulsars, 168 are blazars and 102 are quasars. Figure 8 shows two of the six attributes included in the input of the models alongside with the corresponding error bars. From that figure, one can already observe a good separability of the classes, which improves as the other attributes are taken into account. In general, the pulsars class has larger values of β

---

6. In reality, the FSRQ objects are a type of blazars, even though this is an acronym for Flat-Spectrum Radio Quasars.

7. Specifically, we work with sources whose significance of detection is larger than 30 sigmas, otherwise the classes are too overlapped and some of the inputs have too large error bars.
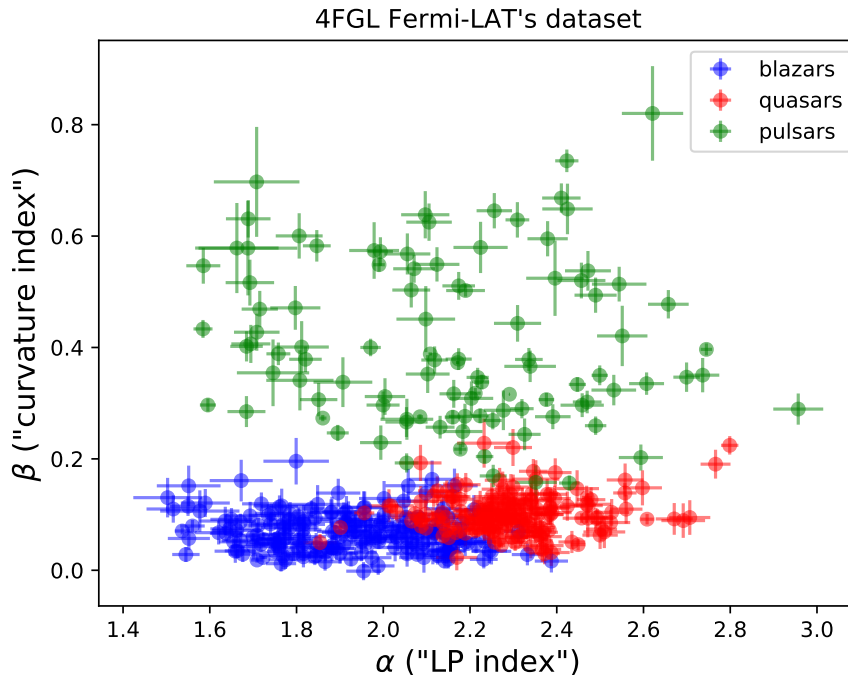
Figure 8: The astrophysics data set considered in this work, showing two of the six attributes included in the input with their associated error bars, for the three classes of sources.

with respect to the other two classes, whereas blazars are separable from quasars in the α direction, but also in the pivot energy direction.

As in the previous experiments, we have estimated the prediction performance of each method on this data set. For this, we have generated 100 splits of the data into training and test partitions with 90% and 10% of the instances, respectively. We have evaluated the test error rate and negative test log-likelihood on each partition, for each method. The average results obtained are displayed in Table 9. As expected, we obtain a significant improvement in the test log-likelihood when using the proposed methods, which improve on the baseline MGP, which does not take into account the input noise. On the other hand, and consistently as well with the previous experiments, the test error rate is similar for all methods, which is an indication that the decision boundaries among the classes are well captured already by the MGP model.

We show in Table 10 the average confusion matrix of the four different methods considered in this work. More concretely, each entry in such a table represents the average (across the different splits) of the number of test samples (which consist of a total of 46 instances) that are assigned a particular class label (rows), for a particular actual label of the example (columns). Therefore, elements outside of the diagonal are classification errors. As can be observed for such table, the errors are more frequent when discriminating between `bll` and `fsrq` classes, than when classifying the `psr` class. This could have been guessed at some

	MGP	NIMGP	NIMGP$_{NN}$	NIMGP$_{FO}$
NLL	0.377±0.0194	**0.246±0.0097**	0.261±0.011	0.292±0.0158
Test error	0.075±0.0043	0.088±0.0038	0.082±0.0041	**0.071±0.0038**

Table 9: Average test error and neg. test log-likelihood (NLL) of each method on the astrophysics data set.

	Model	bll (true)	fsrq (true)	psr (true)
bll (pred.)	**MGP**	17.66±0.31	1.31±0.14	0.01±0.01
	NIMGP	17.61±0.31	1.66±0.12	0.04±0.024
	NIMGP$_{NN}$	17.76±0.3	1.53±0.11	0.01±0.01
	NIMGP$_{FO}$	17.8±0.32	1.27±0.12	0.01±0.01
fsrq (pred.)	**MGP**	1.48±0.13	15.16±0.29	0.36±0.056
	NIMGP	1.53±0.12	14.8±0.29	0.47±0.077
	NIMGP$_{NN}$	1.38±0.12	14.95±0.29	0.46±0.072
	NIMGP$_{FO}$	1.34±0.12	15.2±0.28	0.37±0.06
psr (pred.)	**MGP**	0.00±0.00	0.29±0.054	9.73±0.28
	NIMGP	0.00±0.00	0.3±0.056	9.59±0.27
	NIMGP$_{NN}$	0.00±0.00	0.28±0.047	9.63±0.28
	NIMGP$_{FO}$	0.00±0.00	0.29±0.052	9.72±0.27

Table 10: Average confusion matrix for the astrophysics data set, for each different method considered in this work (see text for details).

extent by looking at Figure 8, where the psr class (green points) is, at least for the two attributes shown there, easier to discriminate than the other two.

6. Conclusions

Multi-class classification problems involve estimating a predictive distribution for the class label given the observed data attributes. Multi-class Gaussian process classifiers are kernel machines that can be used to address these problems with the benefit that they will take into account uncertainty in the estimation process. Often, the supervised machine learning community assumes that the observed data are noise-free in the explaining inputs. Notwithstanding, in some scenarios the measurement of the explaining variables is contaminated with noise. Therefore, input noise is often common in many problems of interest. If this input noise is not modeled correctly, the quality of the resulting predictive distribution can be sub-optimal.

In this paper we have proposed several multi-class GP classifiers that can account for input noise. All these classifiers can be efficiently trained using variational inference to approximate the posterior distribution of the latent variables of the model. They also allow to specify manually, or to infer from the observed data, the level of input noise. Two

approaches are based on introducing extra latent variables in the model to account for noisy inputs, one of them using a neural network to amortize variational parameters. The last method is, however, based on a linear approximation of the GPs of the classifier. Under this approximation input noise is directly translated into output noise.

The inductive bias described is expected to lead to better performance results in practical data sets. To show this, we have evaluated the proposed methods on several experiments, involving synthetic and real data. These include several data sets from the UCI repository, the MNIST data set and also a data set coming from astrophysics. We have compared the results of the proposed methods with those of a standard multi-class GP classifier that ignores input noise. The experiments show that the predictive distribution of the proposed methods is significantly better in terms of the test log-likelihood. The classification error is, however, similar. This means that the decision boundaries of the classifier are not significantly changed. Only the class prediction probabilities.

We have also analyzed the ability of each method to infer the level of input noise from the observed data by approximately maximizing the marginal likelihood. Our results show that accurate estimation results can only be obtained in the case of synthetic data sets, and only for the method that uses a neural network to amortize variational parameters. The other methods prefer input noise variances that are close to zero. This is an unexpected result and analyzing the reason for this behavior is left for future work. In consequence, the method that uses the neural network is the one that obtains the best results in real-world problems in which input noise has been injected in the observed attributes. These problems include data sets extracted from the UCI repository and the MNIST data set.

We have also measured the average time per epoch for each method. Our results show that the method that amortizes the variational parameters using a neural network has a training time that is very similar to that of a standard multi-class GP classifier. This method scales better to bigger data sets. By contrast, the other two methods lead to a significant increment in the computational cost. This increment is much bigger in the case of the method that uses a linear approximation of the GPs employed in the classifier. The reason for this is that computing the required derivatives is expensive.

We have illustrated the utility of a better predictive distribution by carrying out a active learning experiments. In these experiment the uncertainty about the class label of new unseen data is used to choose which data instances should be labeled and introduced into the training set with the goal of improving prediction error the most. Our results show that the method that produces the most accurate predictive distribution (*i.e.*, the one using the neural network to amortize variational parameters) is also the one that identifies the most relevant training points, leading to the largest reduction in the prediction error. This specific application shows that a better predictive distribution can also result in better prediction error.

We have also evaluated the proposed methods in a real data set involving astrophysics data with success, both in terms of error rate and negative test log-likelihood. These experiments add empirical evidence to the hypothesis that if we model input noise the results of multi-class classification using GPs can be enhanced. Summing up, our results indicate that if one is interested in obtaining accurate predictive distributions, it is of vital importance to take into account any potential input noise that has contaminated the explaining variables of the multi-class classification problem.

We believe that the reason for the prediction error being similar is due to the fact that it only depends on the decision boundaries. These boundaries are fully determined by the class label with the largest posterior probability, as estimated by the corresponding method. Therefore, an accurate decision boundary estimation does not require an accurate class posterior probability estimation. However, if one is concerned about the quality of the predictive distribution and the uncertainty in the predictions made by the method, an accurate class posterior probability estimation is strictly required. The proposed methods significantly improve the quality of class posterior probability estimation, while providing similar prediction errors.

Acknowledgments

Appendix A. Stochastic Approximation of the Lower Bound

In this section we describe how to compute an stochastic estimate of the ELBO, described in (10). The stochasticity of the approximation arises from (i) using mini-batches of data to approximate the data-dependent term, and from (ii) approximating the corresponding expectations using Monte Carlo sampling. For this, we use the fact that the approximate distribution q is reparametrizable in the sense that it allows to separate, in each random sample, the dependence on the distribution parameters and the randomness.

Consider the first term in the ELBO and a mini-batch of data instances $\mathcal{B}$. Then,

$$\sum_{i=1}^{N} \mathbb{E}_q \left[\log p(y_i|\mathbf{f}_i) \right] \approx \frac{|\mathcal{B}|}{N} \sum_{i \in \mathcal{B}} \mathbb{E}_q \left[\log p(y_i|\mathbf{f}_i) \right] \tag{17}$$

results in an un-biased estimate of the first term in (10).

Consider now the term $\mathbb{E}_q \left[\log p(y_i|\mathbf{f}_i) \right]$. The expectation with respect to the q distribution can be computed analytically in the case of the random variables $\mathbf{U}$. For this, we only have to use (5). Furthermore, the expectation with respect to the posterior distribution of

$\mathbf{f}_i$ can be approximated accurately using a one-dimensional quadrature. In particular,

$$\mathbb{E}_{q(\mathbf{f}_i)}[\log p(y_i|\mathbf{f}_i)] = (1-e)\frac{1}{S}\sum_{i=1}^{S}\int \mathcal{N}(f^{y_i}(\mathbf{x}_i)|m_{y_i}, v_{y_i})\prod_{c\neq y_i}\Phi\left(\frac{f^{y_i}(\mathbf{x}_i)-m_c}{\sqrt{v_c}}\right)df^{y_i}(\mathbf{x}_i)$$
$$+\frac{e}{C-1}, \tag{18}$$

where $\Phi(\cdot)$ is the cumulative probability of a standard Gaussian distribution and

$$m_c = \mathbf{k}^c_{\mathbf{x}_i,\mathbf{Z}^c}(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c})^{-1}\mathbf{m}_c,$$
$$v_c = k^c_{\mathbf{x}_i,\mathbf{x}_i} - \mathbf{k}^c_{\mathbf{x}_i,\mathbf{Z}^c}(\mathbf{K}^c_{\mathbf{Z}^c,\mathbf{Z}^c})^{-1}\mathbf{K}_{\mathbf{Z}^c,\mathbf{x}_i} + \mathbf{k}^c_{\mathbf{x}_i,\mathbf{Z}^c}(\mathbf{K}^c_{\mathbf{Z}^c,\mathbf{Z}^c})^{-1}\mathbf{S}_c(\mathbf{K}^c_{\mathbf{Z}^c,\mathbf{Z}^c})^{-1}\mathbf{k}^c_{\mathbf{Z}^c,\mathbf{x}_i},$$

for $k = 1,\ldots,C$ with $k^c_{\mathbf{x}_i,\mathbf{x}_i}$ the variance of $f^c(\mathbf{x}_i)$, $\mathbf{k}_{\mathbf{x}_i,\mathbf{Z}^c}$ the covariance vector between $f^c(\mathbf{x}_i)$ and $f^c(\cdot)$ evaluated at $\mathbf{Z}^c$, and $\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}$ the covariance matrix among the values of $f^c(\cdot)$ at $\mathbf{Z}^c$. All these values and matrices can be easily computed given the corresponding covariance functions $\{k_{\theta_c}(\cdot,\cdot)\}_{c=1}^{C}$. Finally, $\mathbf{m}_c$ and $\mathbf{S}_c$ are the mean and covariance parameters of $q(\mathbf{u}^c)$, respectively. See (9) for further details.

It remains now to approximate the expectation of $\mathbb{E}_{q(\mathbf{f}_i)}[\log p(y_i|\mathbf{f}_i)]$ with respect to $q(\mathbf{x}_i)$. This is done by using a Monte Carlo approximation combined with the reparametrization trick (Kingma and Welling, 2014). More precisely, we generate a single sample from $q(\mathbf{x}_i)$ by separating the randomness and the dependence on the parameters of $q(\mathbf{x}_i)$. Namely,

$$\hat{\mathbf{x}}_i = \mathbf{L}_i\boldsymbol{\epsilon} + \boldsymbol{\mu}_i^x, \qquad\qquad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),$$

where $\mathbf{L}_i$ is a diagonal matrix whose entries contain the square root of the diagonal entries of $\mathbf{V}_i^x$, and $\boldsymbol{\mu}_i^x$ is the mean of $\mathbf{x}_i$. See (9) for further details. Let us define $\hat{\mathbf{f}}_i = (f^1(\hat{\mathbf{x}}_i),\ldots,f^C(\hat{\mathbf{x}}_i))^{\mathrm{T}}$. Then,

$$\mathbb{E}_{q(\mathbf{x}_i)}[\mathbb{E}_{q(\mathbf{f}_i)}[\log p(y_i|\mathbf{f}_i)]] \approx \mathbb{E}_{q(\hat{\mathbf{f}}_i)}[\log p(y_i|\hat{\mathbf{f}}_i)],$$

where the right hand side is given by (18) in which we have replaced $\mathbf{x}_i$ by $\hat{\mathbf{x}}_i$. The consequence is that

$$\sum_{i=1}^{N}\mathbb{E}_q\left[\log p(y_i|\mathbf{f}_i)\right] \approx \frac{|\mathcal{B}|}{N}\sum_{i\in\mathcal{B}}\mathbb{E}_{q(\hat{\mathbf{f}}_i)}\left[\log p(y_i|\hat{\mathbf{f}}_i)\right], \tag{19}$$

where the right hand side of is an unbiased estimate of the left hand side.

The second term in (10) can be approximated using a mini-batch and the corresponding expectation can be computed analytically. In particular,

$$\sum_{i=1}^{N}\mathbb{E}_q[\log p(\tilde{\mathbf{x}}_i|\mathbf{x}_i)] \approx \frac{|\mathcal{B}|}{N}\sum_{i\in\mathcal{B}}\mathbb{E}_q[\log p(\tilde{\mathbf{x}}_i|\mathbf{x}_i)]$$
$$= \frac{|\mathcal{B}|}{N}\sum_{i\in\mathcal{B}}\left[-\frac{d}{2}\log 2\pi - \frac{1}{2}\log|\mathbf{V}_i|\right.$$
$$\left. -\frac{1}{2}\text{trace}\left(\mathbf{V}_i(\mathbf{V}_i^x + \boldsymbol{\mu}_i^x(\boldsymbol{\mu}_i^x)^{\mathrm{T}})\right) + \tilde{\mathbf{x}}_i^{\mathrm{T}}\mathbf{V}_i\boldsymbol{\mu}_i^x - \frac{1}{2}\tilde{\mathbf{x}}_i^{\mathrm{T}}\mathbf{V}_i\tilde{\mathbf{x}}_i\right]$$

The third term in (10) is the Kullback-Leibler divergence between Gaussian distributions. This is given by

$$\sum_{c=1}^{C} \mathrm{KL}(q(\mathbf{u}^c)|p(\mathbf{u}^c)) = \sum_{c=1}^{C} \frac{1}{2} \Big[\mathrm{trace}\left((\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{S}_c \right)$$
$$+\mathbf{m}_c^{\mathrm{T}}(\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c)^{-1}\mathbf{m}_c - M + \log \frac{|\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c|}{|\mathbf{S}_c|} \Big] \,.$$

where $\mathbf{m}_c$ and $\mathbf{S}_c$ are the parameters of $q(\mathbf{u}^c)$, and $\mathbf{K}_{\mathbf{Z}^c,\mathbf{Z}^c}^c$ is the covariance matrix of $p(\mathbf{u}^c)$.

The fourth term in (10) can be approximated using a mini-batch and the corresponding Kullback-Leibler divergence can be computed analytically. In particular,

$$\sum_{i=1}^{N} \mathrm{KL}(q(\mathbf{x}^i)|p(\mathbf{x}_i)) \approx \frac{|\mathcal{B}|}{N} \sum_{i \in \mathcal{B}} \mathrm{KL}(q(\mathbf{x}^i)|p(\mathbf{x}_i))$$
$$= \frac{|\mathcal{B}|}{N} \sum_{i \in \mathcal{B}} \frac{1}{2} \Big[\mathrm{trace}\left(\mathbf{V}_i^x \mathbf{I} s^{-1} \right) + (\boldsymbol{\mu}_i^x)^{\mathrm{T}} \boldsymbol{\mu}_i^x s^{-1} - d + \log \frac{s^d}{|\mathbf{V}_i^x|} \Big] \,.$$

Note that all these estimates are unbiased. In practice, the stochastic estimate of the lower bound can be easily codified in a framework such as Tensorflow (Abadi et al., 2015), and the corresponding unbiased estimate of the gradient can be computed using automatic differentiation.

Appendix B. Extra Experiments

In this section we report extra experimental results not covered in the main manuscript.

B.1 Synthetic Experiments Using Data Augmentation and Sampling

In this section we compare the original MGP model when we do not take into account the noise in the inputs with two baselines. Namely, the same model, *i.e.*, MGP trained on an augmented data set with samples from the posterior distribution of the observations $p(\tilde{\mathbf{x}}_i | \mathbf{x}_i) = \mathcal{N}(\tilde{\mathbf{x}}_i | \mathbf{x}_i, \mathbf{V}_i)$ assuming uniform prior. The second baseline is a modified version of MGP in which the input attributes of each batch are obtained by sampling from the corresponding posterior distribution under a uniform prior, ignoring label information. For example, if $x_i \sim N(\tilde{x}_i, \sigma^2)$, where $\tilde{x}_i$ is the noiseless attribute, then $p(\tilde{x}_i | x_i) = \mathcal{N}(\tilde{x}_i | x_i, \sigma^2)$, assuming a uniform prior distribution for $\tilde{x}_i$. When processing each batch of points MGP, we simply sample the input attributes from $p(\tilde{x}_i | x_i)$. The synthetic data set from Section 5.2 is considered. We generate as many data as twice the amount of data instances present in the training set. Note that we cannot sample from the prior distribution of noisy data since we observe noisy observations of the inputs and hence have no access to such a distribution. We report averages over 100 repetitions of the experiments.

	MGP	MGP (aug.)	MGP (sampling)
Noise 0.1	0.76±0.022	0.8±0.022	**0.45±00.009**
Noise 0.25	1.1±0.033	1.2±0.035	**0.61±00.013**
Noise 0.5	1.5±0.042	1.6±0.045	**0.74±00.014**

Table 11: Average Neg. Test Log Likelihood comparison between MGP, MGP with an augmented data set and MGP using sampling from the posterior distribution of the attributes.

	MGP	MGP (aug.)	MGP (sampling)
Noise 0.1	**0.11±0.0032**	0.12±0.0031	0.13±0.0035
Noise 0.25	**0.16±0.0048**	0.17±0.005	0.194±0.0056
Noise 0.5	**0.22±0.0058**	**0.22±0.0063**	0.26±0.0066

Table 12: Average Test Error comparison between MGP, MGP with an augmented data set and MGP using sampling from the posterior distribution of the attributes.

Tables 11 and 12 show the average test log-likelihood and average test error for MGP, MGP on the augmented data set and MGP using the sampling scheme described. We observe that augmenting the data set with samples from the posterior distribution does not improve the results. The sampling approach leads to worse prediction error results. However, the test log-likelihood improves slightly. In any case, the improvements over MGP are smaller than those obtained by the proposed methods NIMGP, NIMGP$_{\text{NN}}$ and NIMGP$_{\text{FO}}$ in Section 5.2. We believe that the reason for the sampling approach not performing very

well compared to these methods is that it ignores the information provided by the labels about the potential vales of the noiseless input attributes.

B.2 Synthetic Experiments Varying the Dimensions and the Number of Points

We repeat the experiments of Section 5.2 changing the number of dimensions d and the number of training data N. We assume the variance of the input noise is given. The results obtained are similar to the ones reported in that Section. Larger d and N lead to better results as there are more explaining attributes and more training data.

	Model	D = 2	D = 5	D = 10
	MGP	0.556±0.018	0.528±0.015	0.273±0.019
Noise 0.1	**NIMGP**	0.36±0.01	0.456±0.012	0.246±0.016
	NIMGP$_{NN}$	0.336±0.01	0.439±0.0096	0.271±0.015
	NIMGP$_{FO}$	0.369±0.012	0.437±0.011	0.244±0.016
	MGP	0.825±0.026	0.674±0.018	0.358±0.022
Noise 0.25	**NIMGP**	0.505±0.012	0.521±0.014	0.309±0.016
	NIMGP$_{NN}$	0.455±0.012	0.501±0.012	0.328±0.016
	NIMGP$_{FO}$	0.519±0.015	0.508±0.013	0.3±0.018
	MGP	1.07±0.033	0.828±0.026	0.419±0.026
Noise 0.5	**NIMGP**	0.62±0.013	0.583±0.016	0.348±0.019
	NIMGP$_{NN}$	0.548±0.013	0.569±0.014	0.353±0.016
	NIMGP$_{FO}$	0.655±0.017	0.565±0.016	0.323±0.02

(a) N=100

	Model	D = 2	D = 5	D = 10
	MGP	0.763±0.025	0.545±0.016	0.261±0.019
Noise 0.1	**NIMGP**	0.27±0.0085	0.292±0.0077	0.154±0.011
	NIMGP$_{NN}$	0.277±0.0089	0.291±0.0076	0.153±0.01
	NIMGP$_{FO}$	0.332±0.011	0.313±0.0078	0.166±0.011
	MGP	1.15±0.036	0.814±0.026	0.442±0.029
Noise 0.25	**NIMGP**	0.391±0.011	0.383±0.011	0.222±0.014
	NIMGP$_{NN}$	0.398±0.012	0.384±0.011	0.218±0.013
	NIMGP$_{FO}$	0.515±0.013	0.413±0.01	0.24±0.014
	MGP	1.41±0.044	1.07±0.031	0.491±0.036
Noise 0.5	**NIMGP**	0.477±0.014	0.463±0.012	0.229±0.016
	NIMGP$_{NN}$	0.5±0.016	0.468±0.012	0.235±0.015
	NIMGP$_{FO}$	0.698±0.021	0.513±0.011	0.26±0.016

(b) N=500

	Model	D = 2	D = 5	D = 10
	MGP	0.802±0.021	0.553±0.02	0.287±0.02
Noise 0.1	**NIMGP**	0.268±0.007	0.252±0.0083	0.141±0.009
	NIMGP$_{NN}$	0.274±0.0077	0.255±0.008	0.139±0.0093
	NIMGP$_{FO}$	0.34±0.009	0.275±0.0081	0.156±0.01
	MGP	1.13±0.036	0.865±0.025	0.454±0.03
Noise 0.25	**NIMGP**	0.372±0.012	0.355±0.0095	0.197±0.012
	NIMGP$_{NN}$	0.388±0.013	0.357±0.0092	0.2±0.012
	NIMGP$_{FO}$	0.529±0.016	0.391±0.0097	0.217±0.013
	MGP	1.49±0.042	1.12±0.037	0.507±0.04
Noise 0.5	**NIMGP**	0.485±0.012	0.437±0.012	0.211±0.015
	NIMGP$_{NN}$	0.507±0.013	0.453±0.012	0.215±0.014
	NIMGP$_{FO}$	0.76±0.021	0.507±0.011	0.238±0.016

(c) N=1000

Table 13: Average negative test log-likelihood and standard deviations over 100 splits on synthetic data sets varying the number of data points and dimensions.

	Model	D = 2	D = 5	D = 10
Noise 0.1	**MGP**	0.146±0.0047	0.179±0.0047	0.0965±0.0069
	NIMGP	0.144±0.0044	0.176±0.005	0.0941±0.0067
	NIMGP$_{NN}$	0.134±0.0044	0.173±0.0041	0.0997±0.0069
	NIMGP$_{FO}$	0.14±0.0046	0.171±0.0045	0.0954±0.0072
Noise 0.25	**MGP**	0.196±0.0059	0.212±0.0061	0.119±0.0078
	NIMGP	0.205±0.0065	0.208±0.006	0.118±0.0072
	NIMGP$_{NN}$	0.186±0.0054	0.199±0.0056	0.122±0.0069
	NIMGP$_{FO}$	0.188±0.0055	0.201±0.0061	0.117±0.0075
Noise 0.5	**MGP**	0.242±0.0072	0.243±0.0043	0.133±0.0092
	NIMGP	0.251±0.0073	0.241±0.0083	0.133±0.0092
	NIMGP$_{NN}$	0.228±0.0066	0.23±0.007	0.129±0.0085
	NIMGP$_{FO}$	0.229±0.0071	0.223±0.0074	0.127±0.009

(a) N=100

	Model	D = 2	D = 5	D = 10
Noise 0.1	**MGP**	0.124±0.0042	0.129±0.0036	0.0663±0.0053
	NIMGP	0.117±0.0039	0.117±0.0031	0.0598±0.0044
	NIMGP$_{NN}$	0.117±0.0037	0.117±0.0031	0.0587±0.0042
	NIMGP$_{FO}$	0.118±0.0039	0.121±0.0031	0.0642±0.005
Noise 0.25	**MGP**	0.177±0.0054	0.169±0.0051	0.099±0.0065
	NIMGP	0.168±0.005	0.154±0.0049	0.0903±0.0059
	NIMGP$_{NN}$	0.167±0.005	0.153±0.005	0.0883±0.0056
	NIMGP$_{FO}$	0.169±0.0051	0.158±0.0051	0.0934±0.0061
Noise 0.5	**MGP**	0.216±0.007	0.209±0.0059	0.102±0.0078
	NIMGP	0.204±0.0068	0.189±0.0054	0.0914±0.0069
	NIMGP$_{NN}$	0.205±0.0069	0.187±0.0052	0.0902±0.0071
	NIMGP$_{FO}$	0.204±0.0068	0.19±0.0054	0.0945±0.0072

(b) N=500

	Model	D = 2	D = 5	D = 10
Noise 0.1	**MGP**	0.122±0.0032	0.115±0.0038	0.0635±0.0043
	NIMGP	0.115±0.003	0.102±0.0035	0.0567±0.0038
	NIMGP$_{NN}$	0.115±0.0031	0.102±0.0033	0.056±0.0039
	NIMGP$_{FO}$	0.116±0.0029	0.107±0.0038	0.0607±0.004
Noise 0.25	**MGP**	0.167±0.0055	0.161±0.0044	0.0897±0.0058
	NIMGP	0.16±0.0052	0.145±0.0043	0.0812±0.0053
	NIMGP$_{NN}$	0.16±0.0049	0.145±0.0041	0.0811±0.0052
	NIMGP$_{FO}$	0.16±0.005	0.148±0.0043	0.0844±0.0054
Noise 0.5	**MGP**	0.213±0.0058	0.193±0.0059	0.0929±0.0074
	NIMGP	0.204±0.0057	0.178±0.0054	0.0852±0.0067
	NIMGP$_{NN}$	0.206±0.0058	0.181±0.0054	0.0844±0.0066
	NIMGP$_{FO}$	0.205±0.0055	0.179±0.0056	0.0865±0.0064

(c) N=1000

Table 14: Average test error and standard deviations over 100 splits on synthetic data sets varying the number of data points and dimensions.

B.3 Synthetic Experiments when Varying Number of Classes

We repeat the experiments of Section 5.2 using a different number of classes. In these experiments we assume the variance of the input noise is given. The results obtained are similar to the ones reported in that section. In general, when we increase the number of class labels the prediction performance is reduced, since the classification problem is more difficult.

	Model	C = 4	C = 5	C = 6	C = 7
	MGP	1.11±0.022	1.28±0.021	1.45±0.02	1.58±0.02
Noise 0.1	**NIMGP**	0.361±0.0064	0.419±0.0075	0.47±0.0067	0.512±0.006
	NIMGP$_{NN}$	0.37±0.0069	0.424±0.007	0.482±0.0074	0.527±0.0064
	NIMGP$_{FO}$	0.45±0.0093	0.512±0.0097	0.583±0.0098	0.636±0.0095
	MGP	1.63±0.027	1.92±0.029	2.13±0.028	2.34±0.027
Noise 0.25	**NIMGP**	0.529±0.0087	0.612±0.0081	0.688±0.0087	0.748±0.0088
	NIMGP$_{NN}$	0.553±0.01	0.637±0.011	0.72±0.01	0.792±0.011
	NIMGP$_{FO}$	0.757±0.014	0.882±0.015	0.976±0.015	1.06±0.015
	MGP	2.13±0.035	2.47±0.04	2.77±0.031	3.06±0.035
Noise 0.5	**NIMGP**	0.685±0.011	0.787±0.01	0.891±0.009	0.991±0.01
	NIMGP$_{NN}$	0.735±0.015	0.842±0.015	0.94±0.013	1.07±0.016
	NIMGP$_{FO}$	1.08±0.019	1.25±0.02	1.37±0.02	1.54±0.017

Table 15: Average negative test log-likelihood error and standard deviations over 100 splits on synthetic data sets varying the number of classes.

	Model	C = 4	C = 5	C = 6	C = 7
	MGP	0.161±0.0029	0.183±0.0031	0.203±0.0029	0.221±0.0026
Noise 0.1	**NIMGP**	0.152±0.003	0.174±0.0029	0.192±0.0029	0.21±0.0028
	NIMGP$_{NN}$	0.153±0.0028	0.174±0.0028	0.192±0.0029	0.211±0.0027
	NIMGP$_{FO}$	0.153±0.003	0.175±0.0028	0.195±0.0028	0.213±0.0026
	MGP	0.228±0.004	0.261±0.0037	0.286±0.0038	0.308±0.0034
Noise 0.25	**NIMGP**	0.219±0.0039	0.251±0.0036	0.277±0.0038	0.297±0.0033
	NIMGP$_{NN}$	0.221±0.004	0.254±0.0039	0.278±0.0037	0.299±0.0035
	NIMGP$_{FO}$	0.221±0.0038	0.252±0.0036	0.279±0.0038	0.299±0.0035
	MGP	0.293±0.005	0.33±0.0052	0.366±0.0044	0.397±0.0046
Noise 0.5	**NIMGP**	0.282±0.0047	0.32±0.0048	0.355±0.0043	0.387±0.0045
	NIMGP$_{NN}$	0.284±0.0052	0.322±0.0052	0.354±0.0046	0.384±0.0044
	NIMGP$_{FO}$	0.281±0.0047	0.318±0.0049	0.354±0.0041	0.385±0.0044

Table 16: Average test error and standard deviations over 100 splits on synthetic data sets varying the number of classes.

B.4 Astrophysics data set without Detection and Curvature Significance

This section reports the results obtained on the Astrophysics data set without using the two attributes described. The results obtained are slightly worse than the ones reported in Section 5.6 which indicates that these attributes contain useful information.

	MGP	**NIMGP**	**NIMGP$_{NN}$**	**NIMGP$_{FO}$**
NLL	0.381±0.021	**0.268±0.013**	0.28±0.012	0.309±0.015
Test error	**0.077±0.0039**	0.0811±0.0043	0.0885±0.0047	0.0867±0.0043

Table 17: Average test error and neg. test log-likelihood (NLL) of each method on the astrophysics data set without using detection and curvature significance attributes.

	Model	bll (true)	fsrq (true)	psr (true)
bll (pred.)	**MGP**	17.52±0.32	1.2±0.11	0.07±0.026
	NIMGP	17.59±0.33	1.45±0.13	0.02±0.014
	NIMGP$_{\textbf{NN}}$	17.43±0.31	1.55±0.13	0.06±0.028
	NIMGP$_{\textbf{FO}}$	17.65±0.32	1.87±0.16	0.02±0.014
fsrq (pred.)	**MGP**	1.59±0.12	15.26±0.29	0.37±0.058
	NIMGP	1.52±0.12	15.08±0.29	0.49±0.07
	NIMGP$_{\textbf{NN}}$	1.65±0.14	14.78±0.28	0.43±0.071
	NIMGP$_{\textbf{FO}}$	1.49±0.11	14.63±0.29	0.37±0.06
psr (pred.)	**MGP**	0.03±0.017	0.3±0.054	9.66±0.27
	NIMGP	0.03±0.017	0.23±0.049	9.59±0.27
	NIMGP$_{\textbf{NN}}$	0.06±0.024	0.43±0.07	9.61±0.28
	NIMGP$_{\textbf{FO}}$	0.00±0.00	0.26±0.05	9.71±0.28

Table 18: Average confusion matrix for the astrophysics data set for the different methods considered in this work without using detection and curvature significance attributes.

Appendix C. Summary of the Parameters of Each Method

We provide a table where we show the parameters used by each of the described methods. We also report the number of parameters employed. All methods share the typical GP hyper-parameters, including the length-scales, amplitudes, noise and inducing points locations. Since this parameters are the same for each method they are not included here. Similarly, all methods share the parameters for the variational distribution $q(\mathbf{u}^c)$ modeling the process values at the inducing points for class c. These parameters are not included here either. NIMGP, NIMGP$_{\text{NN}}$ and NIMGP$_{\text{FO}}$ share the parameters of the variance of the input noise $\mathbf{V}_i$ associated to each training instance. Let N and d be respectively the number of training points and the dimensionality of each method. Last, let M the note the number of parameters (weights and biases) of the neural network considered in NIMGP$_{\text{NN}}$.

	MGP	**NIMGP**	**NIMGP$_{\textbf{NN}}$**	**NIMGP$_{\textbf{FO}}$**
Parameters	None	$\boldsymbol{\mu}_i^x, \mathbf{V}_i^x, \mathbf{V}_i$	$\theta, \mathbf{V}_i$	$\mathbf{V}_i$
# Param.	None	$N \times (d + d + d)$	$M + N \times d$	$N \times d$

Table 19: Parameters and number of parameters of each method

Recall that $\boldsymbol{\mu}_i^x$ and $\mathbf{V}_i^x$ are the variational parameters modeling the posterior distribution (means and variances, respectively) of the attributes of each noiseless data instance. Similarly, θ is the set of parameters of the neural network that predicts $\boldsymbol{\mu}_i^x$ and $\mathbf{V}_i^x$ in NIMGP$_{\text{NN}}$. Finally, $\mathbf{V}_i$ are the input noise variance parameters associated to the i-th data instance. Depending on the number data instances, the methods with the largest number of parameters are either NIMGP or NIMGP$_{\text{NN}}$. In large data sets, however, it is expected that $M \ll N$.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y.Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

S. Abdollahi, F. Acero, M. Ackermann, M. Ajello, W. B. Atwood, M. Axelsson, L. Baldini, J. Ballet, G. Barbiellini, D. Bastieri, J. Becerra Gonzalez, R. Bellazzini, A. Berretta, E. Bissaldi, R. D. Blandford, E. D. Bloom, R. Bonino, E. Bottacini, T. J. Brandt, J. Bregeon, P. Bruel, R. Buehler, T. H. Burnett, S. Buson, R. A. Cameron, R. Caputo, P. A. Caraveo, J. M. Casandjian, D. Castro, E. Cavazzuti, E. Charles, S. Chaty, S. Chen, C. C. Cheung, G. Chiaro, S. Ciprini, J. Cohen-Tanugi, L. R. Cominsky, J. Coronado-Blázquez, D. Costantin, A. Cuoco, S. Cutini, F. D'Ammando, M. DeKlotz, P. de la Torre Luque, F. de Palma, A. Desai, S. W. Digel, N. Di Lalla, M. Di Mauro, L. Di Venere, A. Domínguez, D. Dumora, F. Fana Dirirsa, S. J. Fegan, E. C. Ferrara, A. Franckowiak, Y. Fukazawa, S. Funk, P. Fusco, F. Gargano, D. Gasparrini, N. Giglietto, P. Giommi, F. Giordano, M. Giroletti, T. Glanzman, D. Green, I. A. Grenier, S. Griffin, M. H. Grondin, J. E. Grove, S. Guiriec, A. K. Harding, K. Hayashi, E. Hays, J. W. Hewitt, D. Horan, G. Jóhannesson, T. J. Johnson, T. Kamae, M. Kerr, D. Kocevski, M. Kovac'evic', M. Kuss, D. Landriu, S. Larsson, L. Latronico, M. Lemoine-Goumard, J. Li, I. Liodakis, F. Longo, F. Loparco, B. Lott, M. N. Lovellette, P. Lubrano, G. M. Madejski, S. Maldera, D. Malyshev, A. Manfreda, E. J. Marchesini, L. Marcotulli, G. Martí-Devesa, P. Martin, F. Massaro, M. N. Mazziotta, J. E. McEnery, I. Mereu, M. Meyer, P. F. Michelson, N. Mirabal, T. Mizuno, M. E. Monzani, A. Morselli, I. V. Moskalenko, M. Negro, E. Nuss, R. Ojha, N. Omodei, M. Orienti, E. Orlando, J. F. Ormes, M. Palatiello, V. S. Paliya, D. Paneque, Z. Pei, H. Peña-Herazo, J. S. Perkins, M. Persic, M. Pesce-Rollins, V. Petrosian, L. Petrov, F. Piron, H. Poon, T. A. Porter, G. Principe, S. Rainò, R. Rando, M. Razzano, S. Razzaque, A. Reimer, O. Reimer, Q. Remy, T. Reposeur, R. W. Romani, P. M. Saz Parkinson, F. K. Schinzel, D. Serini, C. Sgrò, E. J. Siskind, D. A. Smith, G. Spandre, P. Spinelli, A. W. Strong, D. J. Suson, H. Tajima, M. N. Takahashi, D. Tak, J. B. Thayer, D. J. Thompson, L. Tibaldo, D. F. Torres, E. Torresi, J. Valverde, B. Van Klaveren, P. van Zyl, K. Wood, M. Yassine, and G. Zaharijas. Fermi large area telescope fourth source catalog. *The Astrophysical Journal Supplement Series*, 247(1):33, 2020. doi: 10.3847/1538-4365/ab6bcb.

A. Armand, D. Filliat, and J. Ibanez-Guzman. Modelling stop intersection approaches using Gaussian processes. In *IEEE Conference on Intelligent Transportation Systems*, pages 1650–1655, 2013.

N. C. Barford. *Experimental Measurements: Precision, Error and Truth*. Wiley, 1985.

S. D. Bay, D. F. Kibler, M. J. Pazzani, and P. Smyth. The UCI KDD archive of large data sets for data mining research and experimentation. *SIGKDD explorations*, 2:81–85, 2000.

H. Bijl, T. B. Schön, J. W. van Wingerden, and M. Verhaegen. System identification through online sparse Gaussian process regression with input noise. *IFAC Journal of Systems and Control*, 2:1–11, 2017.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112:859–877, 2017.

B. Attila Bócsi and L. Csató. Hessian corrected input noise models. In *International Conference on Artificial Neural Networks (ICANN)*, pages 1–8, 2013.

T. D. Bui, D. Hernández-Lobato, J. M. Hernandez-Lobato, Y. Li, and R. E. Turner. Deep Gaussian processes for regression using approximate expectation propagation. In *International Conference on Machine Learning (ICML)*, pages 1472–1481, 2016.

T.D. Bui, J. Yan, and R.E. Turner. A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 18(1):3649–3720, 2017.

W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

Fermi-LAT Collaboration. 4th fgl catalogue. *Available at* `https://heasarc.gsfc.nasa.gov/W3Browse/fermi/fermilpsc.html`, 2019.

Z. Dai, A. Damianou, J. González, and N. Lawrence. Variational auto-encoded deep Gaussian processes. In *International Conference on Learning Representations (ICLR)*, 2016.

P. Dallaire, C. Besse, and B. Chaib-Draa. Learning Gaussian process models from uncertain data. In *International Conference on Neural Information Processing (ICONIP)*, pages 433–440, 2009.

A. Damianou and N. D. Lawrence. Deep Gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

A. Damianou and N. D. Lawrence. Semi-described and semi-supervised learning with Gaussian processes. In *Unvertainty in Artificial Intelligence*, pages 228–237, 2015.

A. Damianou, M. K. Titsias, and N. D. Lawrence. Variational inference for latent variables and uncertain inputs in Gaussian processes. *Journal of Machine Learning Research*, 17:1425–1486, 2016.

M. Deisenroth and C.E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

P. J. Diggle, J. A Tawn, and R. A. Moyeed. Model-based geostatistics. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 47:299–350, 1998.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

H.W. Ge and S.T. Wang. Dependency between degree of fit and input noise in fuzzy linear regression using non-symmetric fuzzy triangular coefficients. *Fuzzy Sets and Systems*, 158:2189–2202, 2007.

A. Girard, C. E. Rasmussen, J. Quiñonero Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 545–552, 2003.

P. W. Goldberg, C. K. I. Williams, and C. M. Bishop. Regression with input-dependent noise: A Gaussian process treatment. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 493–499, 1998.

R. Henao and O. Winther. Predictive active set selection methods for Gaussian processes. *Neurocomputing*, 80:10–18, 2012.

J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*, page 282–290, Arlington, Virginia, USA, 2013.

J. Hensman, A. Matthews, and Z. Ghahramani. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015a.

J. Hensman, A. G. Matthews, M. Filippone, and Z. Ghahramani. MCMC for variationally sparse Gaussian processes. pages 1648–1656, 2015b.

D. Hernández-Lobato, J. M. Hernández-Lobato, and P. Dupont. Robust multi-class Gaussian process classification. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 280–288, 2011.

D. Hernández-lobato, V. Sharmanska, K. Kersting, C. H. Lampert, and N. Quadrianto. Mind the nuisance: Gaussian process classification using privileged noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 837–845. 2014.

M. I. Jordan, Ghahramani Z., T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, (2):183–233, 1999.

K. Kersting, C. Plagemann, P. Pfaff, and W. Burgard. Most likely heteroscedastic Gaussian process regression. In *International conference on Machine learning (ICML)*, pages 393–400, 2007.

H.C. Kim and Z. Ghahramani. Bayesian Gaussian process classification with the EM-EP algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28:1948–1959, 2006.

D. P. Kingma and J. Ba. ADAM: a method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

J. Ko, D. J. Kleint, D. Fox, and D. Haehnelt. Gp-ukf: Unscented kalman filters with Gaussian process prediction and observation models. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1901–1907, 2007.

M. Kuss and C. E. Rasmussen. Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704, 2005.

H.C.L. Law, D. Sutherland, D. Sejdinovic, and S. Flaxman. Bayesian approaches to distribution regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1167–1176, 2018.

M. Lázaro-Gredilla and M. K. Titsias. Variational heteroscedastic Gaussian process regression. In *International Conference on Machine Learning (ICML)*, pages 841–848, 2011.

Q.V. Le, A.J. Smola, and S. Canu. Heteroscedastic Gaussian process regression. In *International Conference on Machine Learning (ICML)*, pages 489–496, 2005.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998.

C. J. Maddison, D. Tarlow, and T. Minka. $A*$ sampling. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 3086–3094. 2014.

A. Mchutchon and C. E. Rasmussen. Gaussian process training with input noise. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 1341–1349, 2011.

T. Minka. Expectation propagation for approximate Bayesian inference. In *Uncertainty in Artificial Intelligence (UAI)*, pages 362–36, 2001.

K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10:1–141, 2017.

K. Murphy. *Machine Learning: a Probabilistic Perspective*. The MIT Press, 2012.

R. M. Neal. Monte Carlo implementation of Gaussian process models for Bayesian regression and classification. Technical report, Department of Statistics, University of Toronto, 1997. URL `http://www.cs.toronto.edu/%7eradford`. Technical Report 9702.

H. Nickisch and C. E. Rasmussen. Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078, 2008.

J. Nogueira, R. Martinez-Cantin, A. Bernardino, and L. Jamone. Unscented Bayesian optimization for safe robot grasping. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 1967–1972, 2016.

J. Oakley and A. O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89:769–784, 2002.

R. Oliveira, L. Ott, V. Guizilini, and F. Ramos. Bayesian optimisation for safe navigation under localisation uncertainty. *arXiv preprint arXiv:1709.02169*, 2017.

A. Panos. *Extreme Multi-label Learning with Gaussian Processes*. PhD thesis, University College London, 2019.

A. Rabaoui, H. Kadri, Z. Lachiri, and N. Ellouze. Using robust features with multi-class SVMs to classify noisy sounds. In *International Symposium on Communications, Control and Signal Processing (ISCCSP)*, pages 594–599, 2008.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

J. Riihimäki, P. Jylänki, and A. Vehtari. Nested expectation propagation for Gaussian process classification with a multinomial probit likelihood. *Journal of Machine Learning Research*, 14:75–109, 2013.

J. A. Sáez, M. Galar, J. Luengo, and F. Herrera. Analyzing the presence of noise in multi-class problems: alleviating its influence with the one-vs-one decomposition. *Knowledge and information systems*, 38(1):179–206, 2014.

M. W. Seeger. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

R. Shu, H. H Bui, S. Zhao, M. J. Kochenderfer, and S. Ermon. Amortized inference regularization. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 4393–4402, 2018.

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 1257–1264, 2006.

D. M. Steinberg and E. V. Bonilla. Extended and unscented Gaussian processes. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 1251–1259, 2014.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 567–574, 2009.

M. Titsias and N. D Lawrence. Bayesian Gaussian process latent variable model. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 844–851, 2010.

M. Van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian processes. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2849–2858, 2017.

C. Villacampa-Calvo and D. Hernández-Lobato. Scalable multi-class Gaussian process classification using expectation propagation. In *International Conference on Machine Learning (ICML)*, pages 3550–3559, 2017.

C. Villacampa-Calvo and D. Hernández-Lobato. Alpha divergence minimization in multi-class Gaussian process classification. *Neurocomputing*, 378:210–227, 2020.

Press W.H., Teukolsky S. A., Vetterling W. T., and Flannery B. P. Numerical recipes: The art of scientific computing (3rd ed.). *Cambridge University Press*, 2007.

C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

A. G. Wilson and Z. Ghahramani. Copula processes. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 2460–2468, 2010.