

Gaussian Process Classification for Variable Fidelity Data

Nikita Klyuchnikov^{a,1}, Evgeny Burnaev^a

^a*Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Building 3,
Moscow 143026, Russia*

Abstract

In this paper we address a classification problem where two sources of labels with different levels of fidelity are available. Our approach is to combine data from both sources by applying a co-kriging schema on latent functions, which allows the model to account item-dependent labeling discrepancy. We provide an extension of Laplace inference for Gaussian process classification, that takes into account multi-fidelity data. We evaluate the proposed method on real and synthetic datasets and show that it is more resistant to different levels of discrepancy between sources than other approaches for data fusion. Our method can provide accuracy/cost trade-off for a number of practical tasks such as crowd-sourced data annotation and feasibility regions construction in engineering design.

Keywords: Gaussian process classification, Variable fidelity data, Laplace inference

1. Introduction

The problem of multi-fidelity modeling [1] arises in the broad range of applied disciplines, such as engineering design, medical diagnostics, and even product development, when an object of interest can be modeled with a cheaper, yet typically less reliable alternative. The main motivation behind multi-fidelity modeling is that low-fidelity data can bring additional benefits in terms of accuracy/cost trade-off, when it is used properly along with high-fidelity data [2, 3]. For example, an article [4] demonstrates that high-quality

*Corresponding author

Email address: nikita.klyuchnikov@skolkovotech.ru (Nikita Klyuchnikov)

linguistic annotation results can be achieved with much lower expenses when non-expert annotators (i.e. low-fidelity data) are employed. The authors concluded that four non-experts per item were enough on average to achieve an expert-level annotation quality for their tasks, although this condition can be relaxed further, by requiring multiple annotations only for a fraction of the dataset. Similarly, in engineering design [5] a high-fidelity source of data can be a physical experiment, whereas a low-fidelity can be a mathematical model or a computer simulation.

Multi-fidelity modeling based on Gaussian processes (GPs) [6] is a reasonable approach for applications discussed above, because of the Bayesian formulation, which allows incorporation of the prior knowledge about the task into the prediction and makes learning on small samples more robust. The latter is especially important, since high-fidelity data typically contains just a few examples. In addition, Gaussian processes are based on kernel functions, whose hyperparameters can be selected via marginal likelihood maximization instead of grid search with cross-validation.

Gaussian process regression for multi-fidelity data has been thoroughly studied in recent years [7, 8], however multi-fidelity classification based on Gaussian processes has been left behind until recently. For example, the work about feasibility regions for aeroelastic stability modeling [9] pointed out that multi-fidelity methods had been limited to continuous response models. Although discrete response models can also be approximated with continuous ones, in some extreme cases, such as binary classification, continuous approximations seem as weird as using Linear regression instead of Logistic regression. On the other hand, developing appropriate models for multi-fidelity classification is essential, because there are problems in engineering design with discrete responses. For instance, report [10] points out the problem of reality gap in robotic simulators and argues the importance of their ability to estimate reliability regions, where accomplishment of actions is accurately predicted by the simulator. This problem has binary responses i.e. success or fail; simulated outcomes of robot’s actions are low-fidelity data, whereas observations of real executions are high-fidelity data. Furthermore, discrete responses are common and convenient when the object of interest is a human. For example, users say they either like a new feature of the product or not during A/B testing, which gives direct evidence of their attitude i.e. high-fidelity data, or users are just asked to imagine the feature and express their preferences during interviewing i.e. low-fidelity data.

In this work, we propose a co-kriging model for latent low- and high- fidelity

functions and extend the Laplace inference algorithm for Gaussian process classification to handle this case. The novelty of our work with respect to other existing ones is adaptation of co-kriging model to classification problem. This model imposes specific dependency and order on sources of data, which help it achieve better performance than more general methods in cases when nature of data is well explained by the model. We evaluate the proposed method on three groups of datasets: artificially generated under the model assumptions, real benchmark datasets with simulated noise for low-fidelity labels and real datasets with true noise. Additionally, contribution of our work includes study of effects of budget distribution among variable fidelity sources under different noise conditions and sensitivity analysis of the proposed model to its hyperparameters.

2. Related work

A comprehensive introduction into GPs in the context of machine learning has been done previously [6]. We were guided by that book during the derivations of our algorithm. More detailed study [11] of methods for approximate binary classification inference based on GPs demonstrates that Laplace Approximation is the fastest inference method with moderate accuracy, whereas Expectation Propagation is the most accurate, but runs approximately 10 times slower. The study outlines that the former should be considered when the error rate is the main metric, although the latter delivers more accurate class probabilities. In addition, when labels contain a lot of noise, the authors outline that all approximation methods tend to produce similar results.

Supervised classification in the presence of noise in labels [12] has been studied with class-conditional random Bernoulli noise, such classification problems have also got theoretical justification of their learnability.

Prior works extensively cover topics connected to multi-task learning [13] in general and multi-output GPs [14] in particular. Multi-fidelity regression based on GPs was also studied in a number of works [15, 16, 17], including a co-kriging setup for fidelities with an exact inference schema for their regression [7]. In our work, we adopt co-kriging for the classification problem by applying this setup on latent functions. Note that there is no exact inference schema for GP classification for single-fidelity case, nor for multi-fidelity one. Several recent works are dedicated to close problems, yet they all consider different aspects. For example, a work on the multivariate generalized linear geostatistical model with spatially structured bias [18] is close to ours,

however, the model studied there doesn't take into account a scaling factor and the proposed inference is confined to MCMC method. A more recent work proposed a framework for handling heterogeneous outputs of GPs with stochastic variational inference [19], also there is a study of the application of heterogeneous multivariate GPs for joint species distribution modeling [20]. Compared to them our work is about a more specific model of multivariate GPs, that can be adapted to a classical algorithmic framework [6] without additional approximation techniques. This tailoring makes our method more robust to noise in labels and accurate than others that use more general models, as we show further in the experimental section.

Heteroscedastic models [21, 22, 23] are complementary to our model in the sense that the former are about modeling input-dependent variance, whereas the latter is about modeling input-dependent bias between low- and high-fidelity processes.

There is a large branch of research on learning from multiple annotators [24], which partially intersects with the applications of our method. Early works in this direction started with different strategies of feature-agnostic labels integration and active learning for optimizing annotation costs [25, 26, 27, 28]. A generative probabilistic model was proposed to estimate annotators expertise along with items annotation difficulty [29], yet features are not observable for the model. Another work [30] studied the problem of pruning low-quality annotators in order to improve the quality of the training set for binary classification problem. The same authors also built an algorithm on top of the SVM, that decreased influence of low-quality entries [31]. Several state-of-the-art works [32, 33, 34] model annotations as random Bernoulli labels dependent on the true class, which in turn is generated via latent Gaussian process; these works have similar setups and provide Variational Bayes and Expectation Propagation inferences for them. Overall, all these works deal with cases when many annotators are available, since otherwise their expertise (fidelity in our case) can barely be resolved. Moreover, [35] showed that without a bit of gold-standard labels, that is, a high-fidelity source, crowd-sourcing labels integration methods will in some cases fail to resolve annotators expertise. Our work stands out from this branch of research due to explicitly fixing fidelities of data sources in our model, which takes into account each item's annotation difficulty.

3. Problem statement

There is a binary function $c : \Omega \rightarrow \{0, 1\}$ defined on the measurable set $\Omega \subset \mathbb{R}^d$. We have two samples:

$$D_H = \{(x_i^H, y_i^H)\}_{i=1}^{n_H} \text{ and } D_L = \{(x_i^L, y_i^L)\}_{i=1}^{n_L}, \quad (1)$$

where $x_i^L, x_i^H \in \Omega$ and $y_i^L, y_i^H \in \{0, 1\}$. Let us also denote $X_L = \{x_i^L\}_{i=1}^{n_L}$, and $X_H = \{x_i^H\}_{i=1}^{n_H}$. All notations are summarized in Table A.4.

Sample D_H contains high-fidelity data, that is, it has much more reliable labels than D_L , which contains low-fidelity data respectively, so its labels can be biased and more noisy. Using the Bayesian approach we formally express this assumption with the following model:

$$\begin{aligned} c(x) &= \mathbb{I}[f_H(x) > 0], \\ p(y_i^H = 1 | f_H(x_i^H)) &= \sigma(f_H(x_i^H)), \\ p(y_i^L = 1 | f_L(x_i^L)) &= \sigma(f_L(x_i^L)), \end{aligned} \quad (2)$$

where \mathbb{I} is an indicator function; $\sigma(z) = \frac{1}{1+\exp(-z)}$ is a sigmoid function; f_L and f_H are Gaussian processes on Ω . In our model we assume these processes are dependent via co-kriging model [7]:

$$f_H(x_i^H) = \rho f_L(x_i^H) + \delta(x_i^H), \quad (3)$$

where $\rho \in \mathbb{R}$ is a linear coefficient, and δ is a residual Gaussian process independent of f_L . Processes f_L and δ have prior kernels k_l and k_d with hyper-parameters θ_l and θ_d respectively. Such dependency between latent processes has been on the one hand acknowledged in many engineering applications [36], on the other hand, it corresponds to the optimal estimate of high-fidelity data according to the Theorem on normal correlation (see [37], theorem 13.1). Parameter ρ can reduce or increase the confidence of the high-fidelity model compared to the low-fidelity one, in particular, $\rho = 1$ corresponds to the case when the low-fidelity source contains high-fidelity labels with additive noise. This parameter is also useful for cases, when low- and high- fidelity labels are mostly opposed to each other. Gaussian process δ can compensate predictions for input-dependent bias in low-fidelity data.

Finally, assuming models (2) and (3) we would like to train a classifier \hat{c} that estimates the function c using samples (1).

4. Solution

For simplicity of notation we omit specifying hyper-parameters (ρ and parameters of kernels θ_l, θ_d) as conditions of probabilities in formulas below.

The predictive distribution of f_H at $x_* \in \Omega$ is:

$$p(f_*^H | D_L, D_H, x_*) = \iint p(f_*^H | \mathbf{f}^L, \boldsymbol{\delta}, X_L, X_H, x_*) p(\mathbf{f}^L, \boldsymbol{\delta} | D_L, D_H) d\mathbf{f}^L d\boldsymbol{\delta}, \quad (4)$$

where

$$\begin{aligned} \boldsymbol{\delta} &= (\delta(x_1^H), \dots, \delta(x_{n_h}^H))^T, \\ \mathbf{f}^L &= (f^L(x_1^L), \dots, f^L(x_{n_L}^L), f^L(x_1^H), \dots, f^L(x_{n_h}^H))^T. \end{aligned}$$

The probability of c to be 1 at point x_* can be expressed by marginalization of the predictive distribution:

$$p(c(x_*) = 1 | D_L, D_H, x_*) = \int \sigma(f_*^H) p(f_*^H | D_L, D_H, x_*) df_*^H. \quad (5)$$

Integrals (4) and (5) don't have analytic solutions, therefore they have to be numerically integrated or approximated analytically. In this work we use Laplace Approximation method to handle the former, whereas the predicted class label based on the latter integral can be easily calculated in the binary case once the predictive distribution is known or estimated [38]:

$$\begin{aligned} \hat{c}(x_*) &= \mathbb{I} \left[\int \sigma(f_*^H) p(f_*^H | D_L, D_H, x_*) df_*^H > \frac{1}{2} \right] = \\ &= \mathbb{I} \left[\int f_*^H p(f_*^H | D_L, D_H, x_*) df_*^H > 0 \right]. \end{aligned}$$

4.1. Laplace Approximation

Prediction based on GPs requires two steps [6]:

1. Obtaining a latent predictive distribution for the test point via marginalizing the posterior distribution over all possible latent values at training points;
2. Marginalizing it over all possible latent values at the test point in order to produce a probabilistic prediction.

Unlike regression problem, where marginalizations are straightforward because all underlying components are Gaussian, prediction of classes is analytically intractable due to non-Gaussian likelihoods.

The idea of Laplace’s method is to handle intractability at step 1 by applying a second order Taylor expansion of posterior’s logarithm around its maximum. Thus we obtain a Gaussian approximation of the posterior distribution, which in turn makes approximate predictive distribution also Gaussian. Next, intractability of step 2 can be resolved by replacing marginalization with maximum a posteriori predictions [38] or approximated with numerical techniques [39, 40].

In the next three sections 4.2, 4.3 and 4.4 we will adjust our solution to fit the algorithmic framework for Laplace Approximation. The key challenge in our case is dependence of y_i^H on multiple latent components, which requires substantial modifications of basic algorithms.

4.2. Mode-fitting

The posterior distribution in integral (4) is approximated with Gaussian distribution $q(\cdot)$:

$$p(\mathbf{f}^L, \boldsymbol{\delta} | D_L, D_H) \approx q(\mathbf{f}^L, \boldsymbol{\delta} | D_L, D_H) = \mathcal{N} \left(\boldsymbol{\xi} = \begin{bmatrix} \mathbf{f}^L \\ \boldsymbol{\delta} \end{bmatrix} \mid \hat{\boldsymbol{\xi}}, \Sigma^{-1} \right), \quad (6)$$

where $\Sigma = -\nabla \nabla \log p(\boldsymbol{\xi} | D_L, D_H) |_{\boldsymbol{\xi}=\hat{\boldsymbol{\xi}}}$ and $\hat{\boldsymbol{\xi}} = \underset{\boldsymbol{\xi}}{\operatorname{argmax}} p(\boldsymbol{\xi} | D_L, D_H)$. Thus, for obtaining approximate posterior distribution we need to calculate these parameters.

According to Bayes formula and monotonic increase of log function, the problem of finding $\hat{\boldsymbol{\xi}}$ is equivalent to:

$$\underset{\boldsymbol{\xi}}{\operatorname{argmax}} p(\boldsymbol{\xi} | D_L, D_H) = \underset{\boldsymbol{\xi}}{\operatorname{argmax}} [\log p(\mathbf{y}^L, \mathbf{y}^H | \boldsymbol{\xi}) + \log p(\boldsymbol{\xi} | X_L, X_H)], \quad (7)$$

where $\mathbf{y}^L = (y_1^L, \dots, y_{n_L}^L)^\top$ and $\mathbf{y}^H = (y_1^H, \dots, y_{n_H}^H)^\top$. Note that the probability of evidence is omitted, since it is independent of the argument. Problem (7) has a unique solution, see details in the Appendix A.1.

Let us now define $\Psi(\boldsymbol{\xi}) \triangleq \log p(\mathbf{y}^L, \mathbf{y}^H | \boldsymbol{\xi}) + \log p(\boldsymbol{\xi} | X_L, X_H)$, and look at its components in more detail. Let also $X = X_L \cup X_H$.

The prior distribution of $\boldsymbol{\xi}$ is normal:

$$p(\boldsymbol{\xi}|X_L, X_H) \sim \mathcal{N}\left(0, K = \begin{bmatrix} k_l(X, X) & 0 \\ 0 & k_d(X_H, X_H) \end{bmatrix}\right). \quad (8)$$

Log-likelihood is:

$$\lambda \triangleq \log p(\mathbf{y}^L, \mathbf{y}^H|\boldsymbol{\xi}) = \sum_{i=1..n_l} \log \sigma(\tilde{y}_i^L f^L(x_i^L)) + \sum_{i=1..n_h} \log \sigma(\tilde{y}_i^H (\rho f^L(x_i^H) + \delta_i)),$$

where for simplicity of notation we use:

$$\delta_i = \delta(x_i^H), \tilde{y}_i^L = (2y_i^L - 1), \text{ and } \tilde{y}_i^H = (2y_i^H - 1).$$

Having figured out expressions for components of $\Psi(\boldsymbol{\xi})$, the solution of problem (7) can be found with iterative Newton's method:

$$\hat{\boldsymbol{\xi}}^{\text{new}} = \hat{\boldsymbol{\xi}}^{\text{old}} - (\nabla \nabla \Psi)^{-1} \nabla \Psi|_{\boldsymbol{\xi}=\hat{\boldsymbol{\xi}}^{\text{old}}}.$$

4.3. Model selection

Let us denote $\tilde{q}(\cdot)$ a Gaussian approximation of the marginal likelihood $p(\mathbf{y}^L, \mathbf{y}^H|X_L, X_H, \rho, \theta_l, \theta_d)$. Model selection implies finding hyper-parameters ρ , θ_l , and θ_d that maximize the approximate log marginal likelihood (this approximation is obtained similarly to the single-fidelity case [6]):

$$\mathcal{L} \triangleq \log \tilde{q}(\mathbf{y}^L, \mathbf{y}^H|X_L, X_H, \rho, \theta_l, \theta_d) = -\frac{1}{2} \hat{\boldsymbol{\xi}}^\top K^{-1} \hat{\boldsymbol{\xi}} + \lambda - \frac{1}{2} \log |B|, \quad (9)$$

where $B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$ and

$$W \triangleq -\nabla \nabla_{\boldsymbol{\xi}} \log p(\mathbf{y}^L, \mathbf{y}^H|\boldsymbol{\xi}) = \begin{bmatrix} A & 0 & 0 \\ 0 & \rho^2 D & \rho D \\ 0 & \rho D & D \end{bmatrix}; \quad (10)$$

$$\begin{aligned} A &= \nabla \nabla_{f^L(X_L)} \lambda = \text{diag} \left(\omega(f^L(x_i^L)) \big|_{i=1..n_l} \right), \\ D &= \nabla \nabla_{\delta} \lambda = \text{diag} \left(\omega(\rho f^L(x_i^H) + \delta_i) \big|_{i=1..n_h} \right), \\ \omega(z) &= \sigma'(z) = \sigma(z)(1 - \sigma(z)). \end{aligned}$$

Unlike single-fidelity case, W in multi-fidelity case is non-diagonal, so computation of its square root is not straightforward. We have derived the exact formula for its fast and numerically stable calculation:

$$W^{\frac{1}{2}} = \begin{bmatrix} A^{\frac{1}{2}} & 0 \\ 0 & \frac{1}{\sqrt{\rho^2+1}} \begin{bmatrix} \rho^2 & \rho \\ \rho & 1 \end{bmatrix} \otimes D^{\frac{1}{2}} \end{bmatrix}, \quad (11)$$

note that matrices A and D are diagonal, so their square roots are easily calculated.

In order to maximize log marginal likelihood (9), one can use gradient-based optimization, which requires its partial derivatives w.r.t. hyper-parameters.

Derivatives of \mathcal{L} and $\hat{\boldsymbol{\xi}}$ w.r.t. kernel hyper-parameters θ_l and θ_d are analogous to formulas in the single-fidelity case ([6], section 5.5.1), thus we omit them here, except the formula 5.23 from [6] for partial derivatives of \mathcal{L} w.r.t. components of $\hat{\boldsymbol{\xi}}$, which reduces calculation of trace to multiplication of i -th diagonal elements. That reduction doesn't take place for multi-fidelity case, since $\frac{\partial W}{\partial \hat{\xi}_i}$ is not diagonal in general. We propose the following modification of that formula:

$$\frac{\partial \mathcal{L}}{\partial \hat{\xi}_i} = -\frac{1}{2} \text{tr} \left((K^{-1} + W)^{-1} \frac{\partial W}{\partial \hat{\xi}_i} \right) = -\frac{1}{2} \sum_{\text{all elements}} \left((K^{-1} + W)^{-1} \circ \frac{\partial W}{\partial \hat{\xi}_i} \right), \quad (12)$$

where \circ is an Hadamard (entrywise) product. Note that $\frac{\partial W}{\partial \hat{\xi}_i}$ is a sparse matrix that has at most 4 non-zero elements (see details in the Appendix A.3), therefore computation time of the derivatives remains linear.

Derivative of \mathcal{L} w.r.t to ρ is:

$$\frac{\partial \mathcal{L}}{\partial \rho} = -\hat{\boldsymbol{\xi}}^\top K^{-1} \frac{\partial \hat{\boldsymbol{\xi}}}{\partial \rho} + \frac{\partial \lambda}{\partial \rho} - \frac{1}{2} \frac{\partial \log |B|}{\partial \rho}. \quad (13)$$

Note that in our setup K doesn't depend on ρ . Now let's look into components of (13) in more detail. We differentiate by ρ the necessary condition of the maximum $\nabla \Psi(\boldsymbol{\xi})|_{\boldsymbol{\xi}=\hat{\boldsymbol{\xi}}} = 0$, where $\nabla \Psi(\boldsymbol{\xi}) = \nabla_{\boldsymbol{\xi}} \lambda - K^{-1} \boldsymbol{\xi}$, obtaining an equation

Table 1: Components of ξ , corresponding derivatives of λ and the explicit term in (14); here $f_i^L = f^L(x_i^L)$ and $f_i^H = \rho f^L(x_i^H) + \delta(x_i^H)$.

components of ξ	components of $\nabla_\xi \lambda$	components of $\frac{\partial \nabla_\xi \lambda _{\xi=\hat{\xi}}}{\partial \rho} \Big _{\text{explicit}}$
$f^L(X_L)$	$y_i^L - \sigma(f_i^L)$	0
$f^L(X_H)$	$\rho(y_i^H - \sigma(f_i^H))$	$y_i^H - \sigma(f_i^H) - \rho f^L(x_i^H) \omega(f_i^H)$
$\delta(X_H)$	$y_i^H - \sigma(f_i^H)$	$-f^L(x_i^H) \omega(f_i^H)$

on ξ :

$$\begin{aligned} \frac{\partial \hat{\xi}}{\partial \rho} &= K \left(-W \frac{\partial \hat{\xi}}{\partial \rho} + \frac{\partial \nabla_\xi \lambda|_{\xi=\hat{\xi}}}{\partial \rho} \Big|_{\text{explicit}} \right) \Rightarrow \\ &\Rightarrow \frac{\partial \hat{\xi}}{\partial \rho} = (I + KW)^{-1} K \left(\frac{\partial \nabla_\xi \lambda|_{\xi=\hat{\xi}}}{\partial \rho} \Big|_{\text{explicit}} \right), \end{aligned} \quad (14)$$

where the components of the *explicit* term in formula (14) and derivatives of λ w.r.t. components of ξ are provided in Table 1.

Next component of (13) is:

$$\frac{\partial \lambda}{\partial \rho} = \sum_{i=1..n_h} \tilde{y}_i^H f^L(x_i^H) (1 - \sigma(\tilde{y}_i^H (\rho f^L(x_i^H) + \delta(x_i^H)))) + \sum_i \frac{\partial \lambda}{\partial \xi_i} \frac{\partial \xi_i}{\partial \rho}. \quad (15)$$

The last component is (see the derivation in the Appendix A.2):

$$\frac{\partial \log |B|}{\partial \rho} = \sum_{\text{all elements}} \left((K^{-1} + W)^{-1} \circ \frac{\partial W}{\partial \rho} \right), \quad (16)$$

where

$$\begin{aligned} \frac{\partial W}{\partial \rho} &= \begin{bmatrix} 0 & 0 \\ 0 & \begin{bmatrix} \rho^2 & \rho \\ \rho & 1 \end{bmatrix} \otimes \frac{\partial D}{\partial \rho} \Big|_{\text{explicit}} + \begin{bmatrix} 2\rho & 1 \\ 1 & 0 \end{bmatrix} \otimes D \end{bmatrix} + \sum_i \frac{\partial W}{\partial \xi_i} \frac{\partial \xi_i}{\partial \rho}; \\ \frac{\partial D}{\partial \rho} \Big|_{\text{explicit}} &= \text{diag} \left(f^L(x_i^H) \zeta(f_i^H) \Big|_{i=1..n_h} \right) \text{ and } \zeta(x) = \sigma''(x). \end{aligned}$$

4.4. Predictions

Once we know estimates of parameters and hyper-parameters, we can use an ordinary schema of exact multi-fidelity posterior from [7] to obtain *MAP predictions*:

$$\mathbb{E}[f_*|D_L, D_H, x_*] \approx \mathbb{E}_q[f_*|D_L, D_H, x_*] = \tilde{k}_*^\top \tilde{K}^{-1} \hat{\mathbf{f}}, \quad (17)$$

where

$$\begin{aligned} \tilde{k}_*^\top &= [k_l(x_*, X_L) \quad \rho k_l(x_*, X_H) + k_d(x_*, X_H)] , \\ \hat{\mathbf{f}} &= [\hat{f}^L(X_L) \quad \hat{f}^L(X_H) \quad \hat{\delta}(X_H)]^\top , \\ \tilde{K} &= \begin{bmatrix} k_l(X_L, X_L) & \rho k_l(X_L, X_H) \\ \rho k_l(X_H, X_L) & \rho^2 k_l(X_H, X_H) + k_d(X_H, X_H) \end{bmatrix} , \\ \hat{\mathbf{f}} &= \begin{bmatrix} \hat{f}^L(X_L) \\ \rho \hat{f}^L(X_H) + \hat{\delta}(X_H) \end{bmatrix} . \end{aligned}$$

5. Experiments

We compared our model with a number of baseline approaches. The baselines are built upon ordinary Gaussian Process Classifier (**gpc**), Logistic Regression (**logit**) and Gradient Boosting Classifier (**xgb**). We trained those baselines in three modes:

1. Training only on high-fidelity data (no prefix);
2. Training on concatenated high- and low- fidelity data (with prefix **C**);
3. Stacking low-fidelity predictions, that is, predictions of a classifier trained on low-fidelity data were used as additional features for training the classifier on high-fidelity data (with prefix **S**).

All GPs-based methods used isotropic RBF kernel. More details regarding the experimental implementation see in the Appendix A.4.

5.1. Evaluation metrics

In order to compare performance of various methods we use areas under receiver operating characteristic curves [41] (ROC AUC) metric.

Further, to aggregate performance across many tests and datasets, we average ROC AUC over them. We also supplement results with figures of

ROC AUC profiles, which show the share of tests where the corresponding methods had greater ROC AUC than the threshold pointed on the abscissa axis. The rule of thumb for assessing such profiles is the higher the curve, the better the corresponding method.

5.2. Datasets

We evaluated models on three groups of datasets:

1. Artificial datasets¹: we constructed datasets by virtue of the model (2) and (3). Latent functions f_L and δ were generated as instances of Gaussian processes, linear coefficients ρ were adjusted to the desired discrepancy (noise level) between low- and high- fidelities. We used input dimensions 2, 5, 10, and 20. For each of them, we generated 10 datasets.
2. Datasets from Penn Machine Learning Benchmarks repository [42]: we selected several representative benchmarks with different types of features, namely `diabetes` (dbts), `german` (grmn), `waveform-40` (wvfr), `satimage` (stmg), `splice` (splc), `spambase` (spmb), `hypothyroid` (hpth), and `mushroom` (mshr). Since some datasets had multiple classes, we also selected one target representative class to test its classification against others: class 0 for `waveform-40` and `splice`, class 1 for `satimage` and class 2 for `diabetes`. Low-fidelity labels were generated by flipping original labels with the specified probability (noise level).
3. Real datasets: we used `music_genre` (mscg) and `sentiment_polarity` (sntp) from [33], which had been annotated with crowd-sourcing. Each object in those datasets was labeled by multiple annotators, therefore we considered majority voting statistic over object labels as high-fidelity and a single random annotation as low-fidelity. Such an approach to model fidelity is reasonable in the context of crowd-sourced annotations where each of them costs some amount of resources (e.g. money or time). For example, some objects are easy to classify with machine learning algorithms, thus one vote would be enough to annotate them, whereas for complex objects many votes are necessary for obtaining good confidence in labels. Finally, since `music_genre` dataset had multiple

¹We published them in this repository <https://github.com/user525/mfgpc>

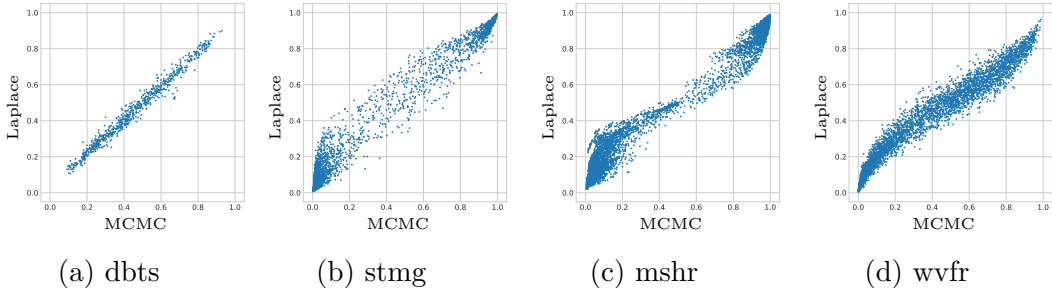


Figure 1: Comparison of predicted class probabilities with multi-fidelity MCMC and Laplace inference on datasets from group 2: typical cases of correlations.

Table 2: Comparison of ROC AUC in a single run for MCMC and Laplace inference on datasets from group 2 during verification tests.

	dbts	grmn	stmg	mshr	splc	spmb	hpth	wvfr
MF gpc Laplace	0.815	0.787	0.998	0.999	0.940	0.942	0.614	0.932
MF gpc MCMC	0.809	0.780	0.997	0.999	0.922	0.946	0.624	0.927

classes, we tested each of them with the one-vs-all scheme as separate datasets.

5.3. Comparison of methods

For datasets in groups 1 and 2 we compared methods with the state-of-the-art hetmogg [19]. For datasets on crowd-sourcing annotation (group 3) we also compared our method with the state-of-the-art method gp-ma [33]. No comparison was made with the method of [34], since we couldn’t find publicly available source code.

At the outset, we verified our implementation of Laplace inference by comparing its predictions with those of MCMC (implemented with PyMC3 package [43]) with the same hyper-parameters on real datasets from group 2 ensuring that true posteriors are non-Gaussian. Each training set contained 75 randomly sampled high fidelity observations and flip probability 0.2 in low-fidelity observations. The typical results of comparison are shown in Figure 1 and Table 2. The overall performance of two inference approaches is on par, whereas correlation behavior resembles patterns observed in single-fidelity GPs classification ([11], figure 6), which lends evidence supporting the correctness of our method.

The main evaluation procedure was the following: for a single test, we

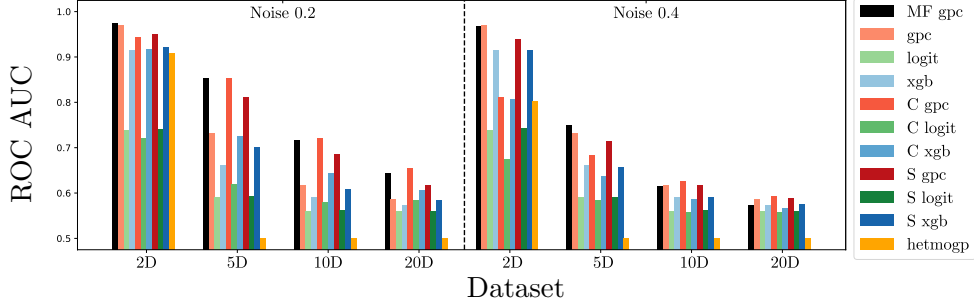


Figure 2: Average ROC AUC among multiple runs on artificial datasets from group 1.

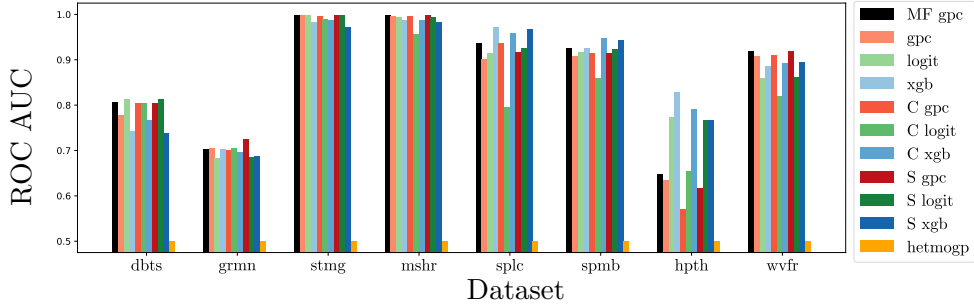


Figure 3: Average ROC AUC among multiple runs on datasets from group 2 with noise level 0.2.

selected a small random subsample of high fidelity observations and 3 times larger subsample of low fidelity observations. We trained all methods on those subsamples and evaluated predictions on the high-fidelity test set. For each dataset, we run 3 tests with different random subsamples, except `sentiment_polarity`, for which we run 15 tests.

We report average ROC AUC across all tests and methods in figures 2, 3, 4 (see also appendix with corresponding tables B.5, B.6, B.7) and table 3. For those tests, each training set contained 75 high fidelity observations. Methods that performed not worse than 1 percent compared to the best result on the dataset are highlighted with bold.

Supplementary ROC AUC profiles are presented in figures 5 and 6. Overall, MF gpc has a good performance, except `sntp` dataset. Notably, on this dataset all GPs-based methods have poor performance, which is not surprising, since we used a translation-invariant isotropic kernel, which is not suited well for highly clustered non-stationary data.

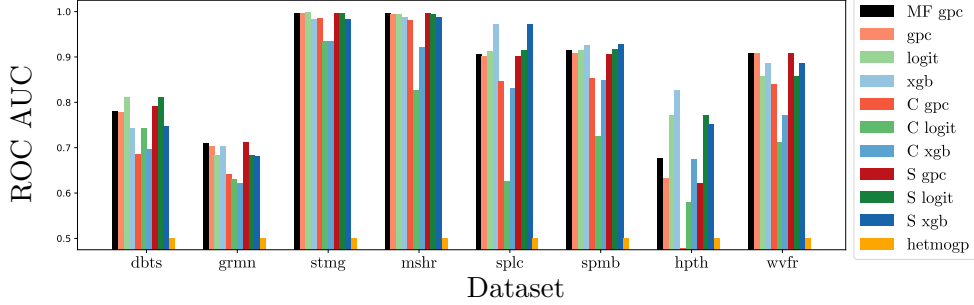


Figure 4: Average ROC AUC among multiple runs on datasets from group 2 with noise level 0.4.

Table 3: Average ROC AUC among multiple runs on datasets from group 3 with natural noise.

	MF gpc	gpc	logit	xgb	C gpc	C logit	C xgb	S gpc	S logit	S xgb	gp-ma
mscg	0.851	0.772	0.794	0.773	0.849	0.812	0.843	0.785	0.797	0.800	0.744
sntp	0.504	0.502	0.542	0.520	0.505	0.569	0.538	0.504	0.553	0.533	0.531

5.4. Budget distribution among variable fidelity sources

We studied how the ratio of low- and high-fidelity samples sizes affects the classification quality of MF gpc on datasets from group 1. An experimental setup was the following: we assumed each high-fidelity entry cost X units, whereas low-fidelity entries cost a fraction of X (with various fractions for different experiments). Training samples were formed based on the total budget: some part of it was allocated for high-fidelity data, the rest was for low-fidelity data. If the whole budget was spent on high-fidelity data, then the training sample contained 100 entries.

Figure 7 demonstrates the more low-fidelity data is available or the less noise is in it, the better classifier works w.r.t. fixed amount of high-fidelity entries. That is, having fixed D_H , adding more data to D_L with the same noise level in low-fidelity labels does not reduce the quality of predictions of our classifier. In the worst-case scenario, when low-fidelity labels are independent of high-fidelity ones, for example they consist of merely random noise, MF gpc model degenerates to an ordinary gpc trained on D_H , because in co-kriging formula (3) component $\rho f_L(x_i^H)$ becomes 0, thus $f_H(x_i^H) = \delta(x_i^H)$.

Figure 8 shows that in case of low noise level in low-fidelity labels sample size advantage overbalances decreased labels quality, thus spending all budget on low-fidelity data is the best option for this case. On the other hand, when

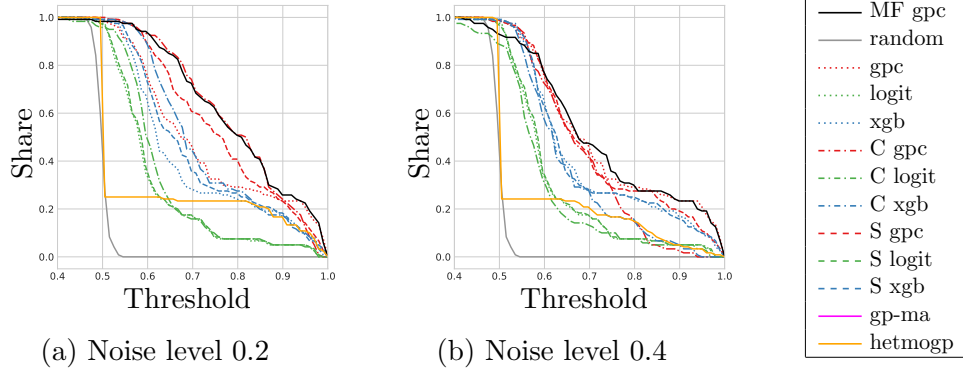


Figure 5: ROC AUC profiles for artificial datasets from group 1.

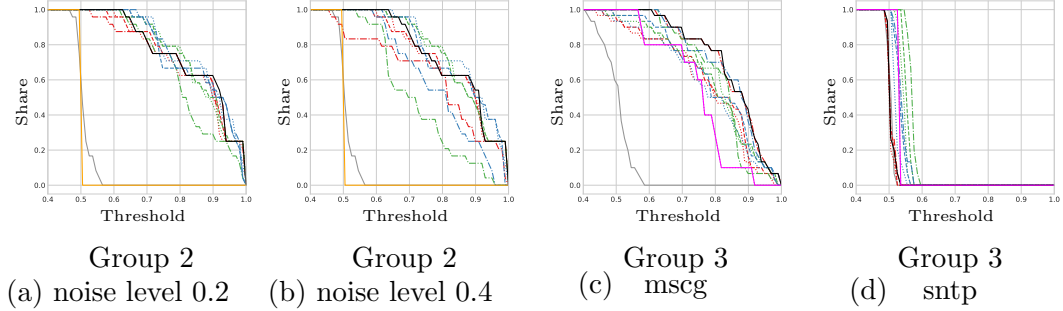


Figure 6: ROC AUC profiles for real datasets from groups 2 and 3. Colors represent the same legend as in figure 5.

the noise in low-fidelity is high, adding any amount of low-fidelity entries instead of high-fidelity ones to training sample reduces the performance of the classifier.

These experiments show that in boundary cases single-fidelity gpc is the choice either for training on low-fidelity data when the noise in labels is low or for training on high-fidelity data when noise is high, whereas MF gpc works slightly better for intermediate noise levels in low-fidelity. It is not trivial to find the right balance in advance, but observations in this section can be used as a rule of thumb in practice.

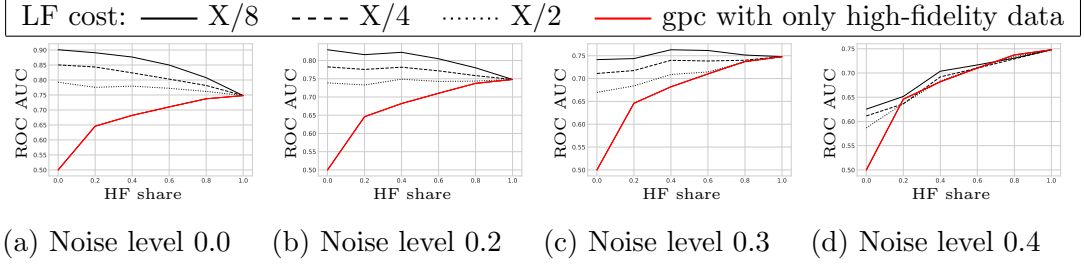


Figure 7: Performance of MF gpc depending on share of budget allocated to high-fidelity data (HF share) for different ratios of low-fidelity cost to high-fidelity cost.

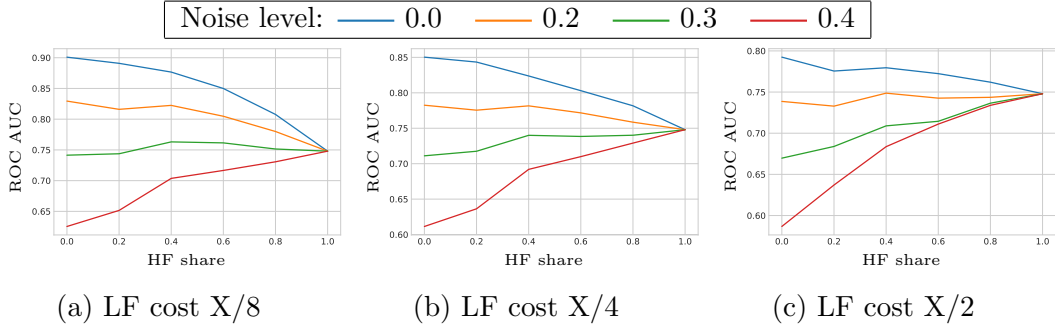


Figure 8: Performance of MF gpc depending on share of budget allocated to high-fidelity data (HF share) for different noise levels in low-fidelity labels.

5.5. Sensitivity to hyperparameters

We used radial basis functions as kernels for Gaussian processes in the following form:

$$k_*(x_i, x_j) = \exp(s_*) \exp\left(-\frac{1}{2} \frac{\|x_i - x_j\|^2}{\sigma_*^2}\right), \quad (18)$$

where $(s_*, \sigma_*) = \theta_*$ are kernel parameters, $* \in \{l, d\}$ indicates the corresponding Gaussian process.

In these series of experiments we first tuned model on training samples, then varied some hyperparameters while kept others fixed to their values obtained during the training. While ρ was varied, parameters of kernels were fixed. While $\theta_l = (s_l, \sigma_l)$ was varied across the grid of s_l and σ_l values, parameters of k_d and ρ were fixed and vice versa for $\theta_d = (s_d, \sigma_d)$. Eventually, for each combination of hyperparameters we estimated model's performance on the corresponding validation samples.

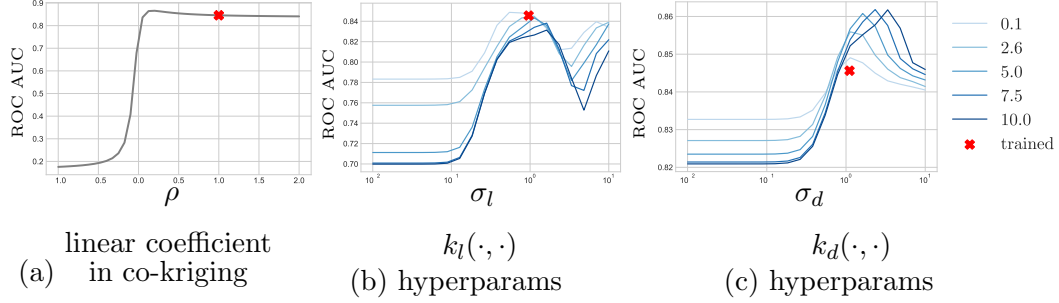


Figure 9: Sensitivity of model performance to its hyperparameters in case of low or moderate noise in low-fidelity labels. Curves of different shades in figures 9b and 9c are associated with the the log-scale coefficient (s_* in (18)) of the corresponding kernel. Red mark indicates parameters and performance of the tuned model during the training.

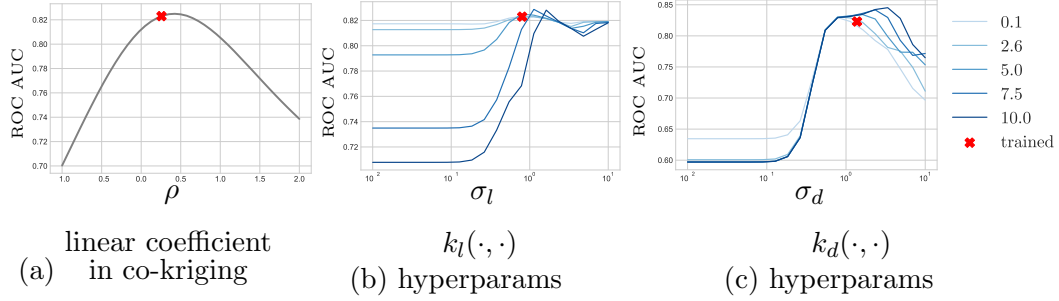


Figure 10: Sensitivity of model performance to its hyperparameters in case of high noise in low-fidelity labels. Curves of different shades in figures 10b and 10c are associated with the the log-scale coefficient (s_* in (18)) of the corresponding kernel. Red mark indicates parameters and performance of the tuned model during the training.

Figures 9 and 10 show a typical sensitivity of model's performance on the validation set with respect to the hyperparameters ρ, θ_l, θ_d for cases with low or moderate noise and case with high noise in low-fidelity data respectively. The former cases are characterized with low local sensitivity to ρ and a sharp decrease in performance when its sign changes; performance is also more sensitive to parameters of k_l than to those of k_d . For latter cases the situation is opposite: the performance is more affected by local changes in ρ , regarding kernels the model is vice versa more sensitive to parameters of k_d than those of k_l .

6. Discussions

Despite the algorithm was proposed for two levels of fidelities, it can be trivially generalized to an arbitrary number of levels assuming the Markov property of fidelity-levels [15].

Further research should be dedicated to the theoretical investigation of budget distribution among fidelities for optimizing the performance of multi-fidelity classifier over single-fidelity ones. The prior work [2] has been successfully studied this issue for the regression problem, which resulted in the analytic formula for the optimal budget balance, although the case for classification problem looks more challenging.

Finally, in order to make our method applicable to a wide range of real projects, its scalability should be improved. A number of approaches to do this has been recently reviewed [8, 44].

7. Conclusions

Multi-fidelity modeling of discrete response surfaces can be put to good use in a number of applied disciplines, yet such methods have got little attention so far. In this work, we extended Laplace inference algorithm for classification based on GPs to make it work with multi-fidelity data. By modeling latent GPs dependency with a co-kriging schema, which has been used previously for multi-fidelity regression, our method can identify not only the overall relevance of low-fidelity data, but resolve local item-dependent discrepancies between fidelities due to inference on residual Gaussian process δ .

We evaluated our method on multiple artificial and real datasets with natural and various levels of simulated noise and compared its performance with a number of baseline approaches and state-of-the-art methods. We also experimentally studied under which conditions adding noisy low-fidelity to the training set increases quality on top of high-fidelity data classification. Depending on the dataset nature, MF gpc can alternate its performance with respect to other methods, however, it is more resistant to different noise levels in low-fidelity labels. That is, when the classifiers based on GPs can learn datasets well, MF gpc has a top performance, whereas in other cases our method is on par with the considered methods.

References

- [1] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, Preprint (2016) 1–57 (2016).
- [2] A. Zaytsev, E. Burnaev, Minimax approach to variable fidelity data interpolation, in: AISTATS, 2017, pp. 652–661 (2017).
- [3] D. J. Toal, Some considerations regarding the use of multi-fidelity kriging in the construction of surrogate models, Struct. Multidiscip. Optim. 51 (6) (2015) 1223–1245 (Jun. 2015).
- [4] R. Snow, B. O’Connor, D. Jurafsky, A. Y. Ng, Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics, 2008, pp. 254–263 (2008).
- [5] W. Pawlus, K. G. Robbersmyr, H. R. Karimi, Mathematical modeling and parameters estimation of a car crash using data-based regressive model approach, Applied Mathematical Modelling 35 (10) (2011) 5091 – 5107 (2011).
- [6] C. E. Rasmussen, C. K. Williams, Gaussian processes for machine learning, MIT Press, 2006 (2006).
- [7] A. I. Forrester, A. Söbester, A. J. Keane, Multi-fidelity optimization via surrogate modelling, in: Proceedings of the royal society of london a: mathematical, physical and engineering sciences, Vol. 463, The Royal Society, 2007, pp. 3251–3269 (2007).
- [8] A. Zaytsev, E. Burnaev, Large scale variable fidelity surrogate modeling, Annals of Mathematics and Artificial Intelligence (Apr 2017).
- [9] C. Dribusch, Multi-fidelity construction of explicit boundaries: Application to aeroelasticity, The University of Arizona, 2013 (2013).
- [10] J.-B. Mouret, K. Chatzilygeroudis, 20 years of reality gap: a few thoughts about simulators in evolutionary robotics, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2017, pp. 1121–1124 (2017).

- [11] H. Nickisch, C. E. Rasmussen, Approximations for binary Gaussian process classification, *Journal of Machine Learning Research* 9 (Oct) (2008) 2035–2078 (2008).
- [12] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, A. Tewari, Learning with noisy labels, in: *Advances in neural information processing systems*, 2013, pp. 1196–1204 (2013).
- [13] Y. Zhang, Q. Yang, A survey on multi-task learning, *arXiv preprint arXiv:1707.08114* (2017).
- [14] M. A. Álvarez, N. D. Lawrence, Computationally efficient convolved multiple output gaussian processes, *Journal of Machine Learning Research* 12 (May) (2011) 1459–1500 (2011).
- [15] M. C. Kennedy, A. O’Hagan, Predicting the output from a complex computer code when fast approximations are available, *Biometrika* 87 (1) (2000) 1–13 (2000).
- [16] L. Le Gratiet, Multi-fidelity gaussian process regression for computer experiments (2013).
- [17] L. Le Gratiet, J. Garnier, Recursive co-kriging model for design of computer experiments with multiple levels of fidelity, *International Journal for Uncertainty Quantification* 4 (5) (2014).
- [18] E. Giorgi, S. S. S. Sesay, D. J. Terlouw, P. J. Diggle, Combining data from multiple spatially referenced prevalence surveys using generalized linear geostatistical models, *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 178 (2) (2015) 445–464 (2015).
- [19] P. Moreno-Muñoz, A. Artés-Rodríguez, M. A. Álvarez, Heterogeneous multi-output gaussian process prediction, *arXiv preprint arXiv:1805.07633* (2018).
- [20] J. Vanhatalo, M. Hartmann, L. Veneranta, Joint species distribution modeling with additive multivariate gaussian process priors and heterogeneous data, *arXiv preprint arXiv:1809.02432* (2018).
- [21] P. W. Goldberg, C. K. Williams, C. M. Bishop, Regression with input-dependent noise: A gaussian process treatment, in: *Advances in neural information processing systems*, 1998, pp. 493–499 (1998).

- [22] Q. V. Le, A. J. Smola, S. Canu, Heteroscedastic gaussian process regression, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 489–496 (2005).
- [23] K. Kersting, C. Plagemann, P. Pfaff, W. Burgard, Most likely heteroscedastic gaussian process regression, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 393–400 (2007).
- [24] J. Zhang, X. Wu, V. S. Sheng, Learning from crowdsourced labeled data: a survey, *Artificial Intelligence Review* 46 (4) (2016) 543–576 (2016).
- [25] V. S. Sheng, F. Provost, P. G. Ipeirotis, Get another label? improving data quality and data mining using multiple, noisy labelers, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 614–622 (2008).
- [26] J. Whitehill, T.-f. Wu, J. Bergsma, J. R. Movellan, P. L. Ruvolo, Whose vote should count more: Optimal integration of labels from labelers of unknown expertise, in: Advances in neural information processing systems, 2009, pp. 2035–2043 (2009).
- [27] B. Mozafari, P. Sarkar, M. J. Franklin, M. I. Jordan, S. Madden, Active learning for crowd-sourced databases, *CoRR* abs/1209.3686 (2012).
- [28] P. G. Ipeirotis, F. Provost, J. Wang, Quality management on amazon mechanical turk, in: Proceedings of the ACM SIGKDD workshop on human computation, ACM, 2010, pp. 64–67 (2010).
- [29] P. Welinder, S. Branson, P. Perona, S. J. Belongie, The multidimensional wisdom of crowds, in: J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23*, Curran Associates, Inc., 2010, pp. 2424–2432 (2010).
- [30] O. Dekel, O. Shamir, Vox populi: Collecting high-quality labels from a crowd, in: Proceedings of the Twenty-Second Annual Conference on Learning Theory, 2009 (June 2009).

- [31] O. Dekel, O. Shamir, Good learners for evil teachers, in: Proceedings of the 26th annual international conference on machine learning, ACM, 2009, pp. 233–240 (2009).
- [32] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, L. Moy, Learning from crowds, *Journal of Machine Learning Research* 11 (Apr) (2010) 1297–1322 (2010).
- [33] F. Rodrigues, F. Pereira, B. Ribeiro, Gaussian process classification and active learning with multiple annotators, in: International Conference on Machine Learning, 2014, pp. 433–441 (2014).
- [34] P. Ruiz, E. Besler, R. Molina, A. K. Katsaggelos, Variational Gaussian process for missing label crowdsourcing classification problems, in: Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on, IEEE, 2016, pp. 1–6 (2016).
- [35] N. D. Penna, M. D. Reid, Crowd & prejudice: An impossibility theorem for crowd labelling without a gold standard (2012). [arXiv:arXiv:1204.3511](#).
- [36] A. I. J. Forrester, A. Sbester, A. J. Keane, Engineering Design via Surrogate Modelling, John Wiley & Sons, Ltd, 2008 (jul 2008). doi: 10.1002/9780470770801.
- [37] R. S. Liptser, A. N. Shiryaev, Statistics of Random Processes, Springer Berlin Heidelberg, 2001 (2001). doi:10.1007/978-3-662-10028-8.
- [38] C. M. Bishop, Neural networks for pattern recognition, Oxford university press, 1995 (1995).
- [39] D. J. MacKay, The evidence framework applied to classification networks, *Neural computation* 4 (5) (1992) 720–736 (1992).
- [40] C. K. Williams, D. Barber, Bayesian classification with Gaussian processes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (12) (1998) 1342–1351 (1998).
- [41] J. A. Hanley, B. J. McNeil, The meaning and use of the area under a receiver operating characteristic (roc) curve., *Radiology* 143 (1) (1982) 29–36 (1982).

- [42] R. S. Olson, W. La Cava, P. Orzechowski, R. J. Urbanowicz, J. H. Moore, PMLB: a large benchmark suite for machine learning evaluation and comparison, *BioData Mining* 10 (1) (2017) 36 (Dec 2017).
- [43] J. Salvatier, T. V. Wiecki, C. Fonnesbeck, Probabilistic programming in python using pymc3, *PeerJ Computer Science* 2 (2016) e55 (2016).
- [44] H. Liu, Y.-S. Ong, X. Shen, J. Cai, When gaussian process meets big data: A review of scalable gps, *arXiv preprint arXiv:1807.01065* (2018).
- [45] K. Schäcke, *On the Kronecker Product* (2013).

Appendix A. Theoretical and experimental details

This appendix contains detailed information about some key identities and experimental setup. We also published source code for our model and experiments in this repository <https://github.com/user525/mfgpc>.

Notations	Descriptions	Specification
Ω	the measurable domain of data	$\Omega \subset \mathbb{R}^d$
$c(\cdot)$	a binary function defined on Ω	-
D_H	high-fidelity sample	$D_H = \{(x_i^H, y_i^H)\}_{i=1}^{n_H}$
D_L	low-fidelity sample	$D_L = \{(x_i^L, y_i^L)\}_{i=1}^{n_L}$
X_H	points of high-fidelity sample	$X_H = \{x_i^H\}_{i=1}^{n_H}$
X_L	points of low-fidelity sample	$X_L = \{x_i^L\}_{i=1}^{n_L}$
$f_H(\cdot)$	latent Gaussian Process for high-fidelity	$f_H(x) = \rho f_L(x) + \delta(x)$
$f_L(\cdot)$	latent Gaussian Process for low-fidelity	-
$\delta(\cdot)$	latent residual Gaussian Process	-
ρ	linear coefficient for co-kriging dependency	$\rho \in \mathbb{R}$
$k_l(\cdot, \cdot)$	prior kernel for $f_L(\cdot)$	-
$k_d(\cdot, \cdot)$	prior kernel for $\delta(\cdot)$	-
θ_l	parameters of kernel k_l	a multi-dimensional real vector
θ_d	parameters of kernel k_d	a multi-dimensional real vector
$\sigma(\cdot)$	sigmoid function	$\sigma(z) = \frac{1}{1+\exp(-z)}$
$\omega(\cdot)$	first derivative of $\sigma(\cdot)$	$\omega(z) = \sigma(z)(1 - \sigma(z))$
$\zeta(\cdot)$	second derivative of $\sigma(\cdot)$	$\zeta(x) = \sigma(x)(1 - \sigma(x))(1 - 2\sigma(x))$
λ	log-likelihood	$\log p(\mathbf{y}^L, \mathbf{y}^H \boldsymbol{\xi})$
\mathcal{L}	approximate log marginal likelihood	$\log \tilde{q}(\mathbf{y}^L, \mathbf{y}^H X_L, X_H, \rho, \theta_l, \theta_d)$

Table A.4: Some of notations used in the paper.

Appendix A.1. Correctness Of The Method

Optimization problem (7) has a unique solution if Ψ is concave. We prove it by showing that Hessian of Ψ is negative semi-definite. The Hessian is

$$\nabla \nabla \Psi(\boldsymbol{\xi}) = -W - K^{-1}, \quad (\text{A.1})$$

where K is positive semi-definite, since it is a kernel matrix.

Matrices A and D in the definition of W (10) are positive semi-definite, because their diagonal elements are non-negative. The block of W that contains D can be represented via Kronecker product:

$$\begin{bmatrix} \rho^2 D & \rho D \\ \rho D & D \end{bmatrix} = \begin{bmatrix} \rho^2 & \rho \\ \rho & 1 \end{bmatrix} \otimes D. \quad (\text{A.2})$$

Both multiplicands in (A.2) are positive semi-definite, thus, their Kronecker product is also positive semi-definite [45].

Hence, matrix W is positive semi-definite, because it factorizes into two positive semi-definite blocks. Finally, the Hessian is negative semi-definite as a negation of sum of two positive semi-definite matrices.

Appendix A.2. Inference For Equation (16)

$$\begin{aligned}
\frac{\partial \log |B|}{\partial \rho} &= \text{tr} \left(B^{-1} \frac{\partial B}{\partial \rho} \right) = \text{tr} \left(B^{-1} \left(\frac{\partial W^{\frac{1}{2}}}{\partial \rho} K W^{\frac{1}{2}} + W^{\frac{1}{2}} K \frac{\partial W^{\frac{1}{2}}}{\partial \rho} \right) \right) = \\
&= \text{tr} \left(B^{-1} \left(W^{-\frac{1}{2}} W^{\frac{1}{2}} \right) \frac{\partial W^{\frac{1}{2}}}{\partial \rho} K W^{\frac{1}{2}} \right) + \text{tr} \left(B^{-1} W^{\frac{1}{2}} K \frac{\partial W^{\frac{1}{2}}}{\partial \rho} \left(W^{\frac{1}{2}} W^{-\frac{1}{2}} \right) \right) = \\
&= \text{tr} \left(K W^{\frac{1}{2}} B^{-1} W^{-\frac{1}{2}} \left(W^{\frac{1}{2}} \frac{\partial W^{\frac{1}{2}}}{\partial \rho} \right) \right) + \text{tr} \left(W^{-\frac{1}{2}} B^{-1} W^{\frac{1}{2}} K \left(\frac{\partial W^{\frac{1}{2}}}{\partial \rho} W^{\frac{1}{2}} \right) \right) = \\
&= \text{tr} \left((K^{-1} + W)^{-1} \frac{\partial W}{\partial \rho} \right) = \sum_{\text{all elements}} \left((K^{-1} + W)^{-1} \circ \frac{\partial W}{\partial \rho} \right)
\end{aligned} \tag{A.3}$$

The last line of (A.3) is obtained because of the following identities:

$$\frac{\partial W^{\frac{1}{2}}}{\partial \rho} W^{\frac{1}{2}} + W^{\frac{1}{2}} \frac{\partial W^{\frac{1}{2}}}{\partial \rho} = \frac{\partial W^{\frac{1}{2}} W^{\frac{1}{2}}}{\partial \rho} = \frac{\partial W}{\partial \rho} \tag{A.4}$$

$$K W^{\frac{1}{2}} B^{-1} W^{-\frac{1}{2}} = K \left(W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \right) W^{-1} = K (K + W^{-1})^{-1} W^{-1} = (K^{-1} + W)^{-1} \tag{A.5}$$

$$W^{-\frac{1}{2}} W^{\frac{1}{2}} B^{-1} K = W^{-1} \left(W^{\frac{1}{2}} B^{-1} W^{\frac{1}{2}} \right) K = W^{-1} (K + W^{-1})^{-1} K = (K^{-1} + W)^{-1} \tag{A.6}$$

Appendix A.3. Components Of (12)

Let us denote $M \triangleq (K^{-1} + W)^{-1}$.

For indices i corresponding to low-fidelity data on X_L ($i = 1 \dots n_l$):

$$M_{i,i} \frac{\partial^3}{\partial \hat{\xi}_i^3} \lambda \equiv M_{i,i} \zeta(f_L(x_i^L)) \quad (\text{A.7})$$

For indices i corresponding to low-fidelity data on X_H ($i = n_l + 1 \dots n_l + n_h$):

$$\begin{aligned} & \left(M_{i,i} \frac{\partial^3}{\partial \hat{\xi}_i^3} + 2M_{i,i+n_h} \frac{\partial^3}{\partial \hat{\xi}_{i+n_h} \partial \hat{\xi}_i^2} + M_{i+n_h,i+n_h} \frac{\partial^3}{\partial \hat{\xi}_{i+n_h}^2 \partial \hat{\xi}_i} \right) \lambda \equiv \\ & \equiv (M_{i,i} \rho^3 + 2M_{i,i+n_h} \rho^2 + M_{i+n_h,i+n_h} \rho) \zeta(\rho f^L(x_{i-n_l}^H) + \delta_{i-n_l}) \end{aligned} \quad (\text{A.8})$$

For indices i corresponding to delta on X_H ($i = n_l + n_h + 1 \dots n_l + 2n_h$):

$$\begin{aligned} & \left(M_{i,i} \frac{\partial^3}{\partial \hat{\xi}_i^3} + 2M_{i,i-n_h} \frac{\partial^3}{\partial \hat{\xi}_{i-n_h} \partial \hat{\xi}_i^2} + M_{i-n_h,i-n_h} \frac{\partial^3}{\partial \hat{\xi}_{i-n_h}^2 \partial \hat{\xi}_i} \right) \lambda \equiv \\ & \equiv (M_{i,i} + 2M_{i,i-n_h} \rho + M_{i-n_h,i-n_h} \rho^2) \zeta(\rho f^L(x_{i-n_l-n_h}^H) + \delta_{i-n_l-n_h}) \end{aligned} \quad (\text{A.9})$$

Appendix A.4. Specifications Of Implementation

For experiments we used Python 3.6.

- Implementations of Gaussian Process Classifiers and Logistic Regressions were used from scikit-learn package²; Out method was implemented on top of `GaussianProcessClassifier` module from this package;
- Classifiers based on GPs used isotropic RBF kernels;
- Classifiers based on GPs and Logistic Regression were used in the pipeline with the Standard Scaler features preprocessor;
- Implementation of Gradient Boosting Classifier was used from XGBoost module³ with the following parameters: `n_estimators=100`, `max_depth=3`, `learning_rate=0.05`, `subsample=0.85`;
- For each run of the evaluation procedure we generated a random training subsample that has at least one label of each class (both for low- and high-fidelity subsamples), that is, positive and negative;
- The set of random seeds for different runs was shared across series of method-dataset evaluations.

²<http://scikit-learn.org/>

³<https://xgboost.readthedocs.io>

Appendix B. Supplementary materials

This appendix contains supplementary materials for experimental results.

Table B.5: Average ROC AUC among multiple runs on artificial datasets from group 1.

Noise level	0.2				0.4			
Dimensionality	2D	5D	10D	20D	2D	5D	10D	20D
MF gpc	0.975	0.853	0.716	0.643	0.968	0.750	0.615	0.573
gpc	0.970	0.732	0.616	0.587	0.970	0.732	0.616	0.587
logit	0.738	0.590	0.559	0.559	0.738	0.590	0.559	0.559
xgb	0.914	0.662	0.591	0.574	0.914	0.662	0.591	0.574
C gpc	0.944	0.854	0.721	0.654	0.811	0.683	0.626	0.592
C logit	0.721	0.619	0.580	0.585	0.675	0.584	0.557	0.557
C xgb	0.916	0.725	0.644	0.607	0.807	0.637	0.586	0.567
S gpc	0.949	0.812	0.686	0.616	0.938	0.713	0.617	0.589
S logit	0.740	0.592	0.563	0.559	0.742	0.591	0.561	0.559
S xgb	0.921	0.700	0.608	0.583	0.914	0.657	0.591	0.575
hetmogp	0.909	0.500	0.500	0.500	0.802	0.500	0.500	0.500

Table B.6: Average ROC AUC among multiple runs on datasets from group 2 with noise level 0.2.

	dbts	grmn	stmg	mshr	splc	spmb	hpth	wvfr
MF gpc	0.805	0.702	0.997	0.997	0.936	0.925	0.646	0.919
gpc	0.778	0.704	0.997	0.995	0.901	0.907	0.633	0.908
logit	0.812	0.683	0.998	0.994	0.913	0.915	0.772	0.858
xgb	0.742	0.702	0.982	0.987	0.971	0.925	0.827	0.886
C gpc	0.804	0.699	0.996	0.995	0.937	0.914	0.570	0.910
C logit	0.803	0.704	0.989	0.955	0.794	0.859	0.654	0.820
C xgb	0.767	0.696	0.987	0.987	0.958	0.946	0.791	0.891
S gpc	0.804	0.725	0.997	0.997	0.915	0.914	0.616	0.918
S logit	0.812	0.684	0.997	0.994	0.924	0.923	0.766	0.861
S xgb	0.738	0.687	0.971	0.983	0.967	0.943	0.766	0.895
hetmogp	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500

Table B.7: Average ROC AUC among multiple runs on datasets from group 2 with noise level 0.4.

	dbts	grmn	stmng	mshr	splc	spmb	hpth	wvfr
MF gpc	0.781	0.710	0.997	0.996	0.905	0.914	0.676	0.909
gpc	0.778	0.704	0.997	0.995	0.901	0.907	0.633	0.908
logit	0.812	0.683	0.998	0.994	0.913	0.915	0.772	0.858
xgb	0.742	0.702	0.982	0.987	0.971	0.925	0.827	0.886
C gpc	0.685	0.642	0.986	0.981	0.846	0.852	0.479	0.840
C logit	0.743	0.630	0.934	0.827	0.626	0.725	0.579	0.711
C xgb	0.697	0.621	0.934	0.921	0.831	0.849	0.674	0.771
S gpc	0.791	0.711	0.997	0.996	0.901	0.906	0.622	0.907
S logit	0.811	0.683	0.997	0.994	0.914	0.916	0.771	0.858
S xgb	0.747	0.680	0.984	0.988	0.972	0.927	0.752	0.885
hetmogp	0.500	0.500	0.500	0.500	0.500	0.500	0.500	0.500