Pattern Recognition 88 (2019) 298–311

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Learning from crowds with variational Gaussian processes[☆]

Pablo Ruiz[a,1,*], Pablo Morales-Álvarez[b,1,*], Rafael Molina[b], Aggelos K. Katsaggelos[a]

[a] Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA
[b] Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain

ARTICLE INFO

Article history:
Received 26 February 2018
Revised 21 August 2018
Accepted 17 November 2018
Available online 20 November 2018

Keywords:
Crowdsourcing
Classification
Gaussian processes
Bayesian modeling
Variational inference

ABSTRACT

Solving a supervised learning problem requires to label a training set. This task is traditionally performed by an expert, who provides a label for each sample. The proliferation of social web services (e.g., Amazon Mechanical Turk) has introduced an alternative crowdsourcing approach. Anybody with a computer can register in one of these services and label, either partially or completely, a dataset. The effort of labeling is then shared between a great number of annotators. However, this approach introduces scientifically challenging problems such as combining the unknown expertise of the annotators, handling disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators. All these problems require probabilistic sound solutions which go beyond the naive use of majority voting plus classical classification methods. In this work we introduce a new crowdsourcing model and inference procedure which trains a Gaussian Process classifier using the noisy labels provided by the annotators. Variational Bayes inference is used to estimate all unknowns. The proposed model can predict the class of new samples and assess the expertise of the involved annotators. Moreover, the Bayesian treatment allows for a solid uncertainty quantification. Since when predicting the class of a new sample we might have access to some annotations for it, we also show how our method can naturally incorporate this additional information. A comprehensive experimental section evaluates the proposed method with synthetic and real experiments, showing that it consistently outperforms other state-of-the-art crowdsourcing approaches.

1. Introduction

The main goal in supervised learning is to find a mapping that predicts labels from features [1–3]. Most of the works in supervised learning assume that the training samples have been labeled with no errors by an expert [4,5]. However, the recent advent of social web services has introduced a new approach to address the labeling problem. The term *crowdsourcing* was coined in 2006 by J. Howe [6] to describe *"the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined generally large group of people in the form of an open call"*.

In the last decade, many crowdsourcing services have proliferated in the Internet, where a dataset can be published and millions of people around the world can provide labels in exchange for a reward [7]. Amazon Mechanical Turk (www.amt.com), Galaxy Zoo (www.galaxyzoo.org), Zooniverse (www.zooniverse.org), Crowdflowers (www.crowdflower.com) or Clickworker (www.clickworker.com) are among the most popular ones. Due to the great number of potential annotators, large data sets can be labeled in a very short time. However, this approach introduces new challenging problems: combining the unknown expertise of annotators, dealing with disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators [7].

The first paper on crowdsourcing dates back to 1979 [8]. Early contributions attempted to estimate the underlying true labels and the reliability of the annotators, but were not conceived to learn a classifier. This idea was explored by Raykar et al. [9], who proposed to jointly estimate the coefficients of a logistic regression (LR) classifier and the annotators' expertise. The latter is modeled through the *sensitivity* and *specificity* concepts, which refer to the accuracy of the annotator when labelling instances from each class. Yan et al. [10] (see also the subsequent journal version [11]), introduced a crowdsourcing classifier (also based on LR) which considers a feature-dependent model for the annotators' expertise. The main limitation of these two approaches is the simple LR classification model, which can only deal with linearly sep-

[☆] This work was supported by the Spanish Ministry of Economy and Competitiveness under project DPI2016-77869-C2-2-R, the US Department of Energy (DE-NA0002520) and the Visiting Scholar Program at the University of Granada. PMA received financial support through *La Caixa* Fellowship for Doctoral Studies (*La Caixa* Banking Foundation, Barcelona, Spain).

[*] Corresponding authors.

E-mail addresses: mataran@northwestern.edu (P. Ruiz), pablomorales@decsai.ugr.es (P. Morales-Álvarez), rms@decsai.ugr.es (R. Molina), aggk@eecs.northwestern.edu (A.K. Katsaggelos).

[1] The first two authors contributed equally.

https://doi.org/10.1016/j.patcog.2018.11.021
0031-3203/© 2018 Elsevier Ltd. All rights reserved.

arable data. Rodrigues et al. [12] overcame this problem by introducing a crowdsourcing classifier based on Gaussian Processes (GP) [13–15]. GP theory makes use of the so-called *"kernel trick"* [1, Chapter 6] to model complex classification problems where the decision boundary may be non-linear. Expectation Propagation (EP) [13, Section 3.6] is used as inference procedure in [12]. To the best of our knowledge, this is the most recent general-purpose probabilistic crowdsourcing approach (see also [16, Section 2.2].)

Nowadays, crowdsourcing is a really active and promising research field, in which these general-purpose crowdsourcing methods are being tailored to a wide range of relevant problems (see the recent survey [7] and related works [17,18]). Crowdsourcing is being applied to modern areas such as ecological monitoring and conservation [19], plant phenotyping [20,21], remote sensing [22], mitosis detection in breast cancer histology images [23], topic modeling from crowds [16], and detection of glitches in signals acquired by the laureate Laser Interferometer Gravitational-Wave Observatory (LIGO) [24]. Moreover, there exist some recent attempts to combine crowdsourcing with Deep Learning approaches [23,25], and new challenges, such as the optimal expert validation of the crowdsourced labels [26], are emerging.

In this work we address the crowdsourcing classification problem. As in [12], the true underlying training labels are modeled as latent variables by means of a GP. A sensitivity-specificity model is used for the annotators (as in [9] and [12]). However, there exist two main differences with [12]: 1) we use Variational Bayes inference (VB) to estimate all unknowns (instead of EP), and 2) we model sensitivity and specificity as stochastic variables (instead of point parameters). Several reasons motivated our choice of Variational inference. First, it is well-known that the EP iterative procedure does not guarantee convergence, and it may not be able to capture complex posterior distributions (e.g., multi-modal) [1, Section 10.7]. Second, as it will become clear in the experiments, the EP inference is usually slower in practice (which, in fact, has led to the introduction of some strategies to optimize it [27]). A thorough experimentation (including comparisons with the aforementioned approaches in [9,10] and [12] among others) will show that the proposed ideas can contribute to advance the current state-of-the-art in crowdsourcing classification. Moreover, the proposed model naturally lends itself to the integration of annotations that may have been provided for test instances in the prediction of their true class. The experiments will show that, if test annotations are available, this hybrid human-machine prediction is significantly more accurate than the one produced by either the machine or the annotators alone. To the best of our knowledge, this extension had not been addressed in any previous work.

This paper gathers together, clarifies, and significantly extends the ideas in our two conference contributions [28,29]. The main novelties are: first, sensitivity and specificity are treated as stochastic variables (they are estimated through non-degenerate posterior distributions instead of point estimates). This allows for a better uncertainty quantification and, thus, an enhancement in the experimental results. Second, we show how our model can naturally integrate in the prediction annotations that may have been provided for test instances. If there are no such annotations, the new predictive distribution recovers the old one. Third, the experiments are exhaustively extended in several ways: the new methodology to integrate test set annotations is evaluated, a new type of data popular in crowdsourcing is introduced (*semi-synthetic* data), the computational cost is assessed, and the annotators' expertise estimations are reported in all experiments. Fourth, the experimental section does not restrict itself to the performance of the proposed method, but also examines the behavior of other state-of-the-art approaches that it is compared against. Thus, it can be useful as a brief experimental review of the main current crowdsourcing methods.

The rest of the paper is organized as follows. To facilitate the reading of the paper, an exhaustive glossary of all the symbols used in this work is included in Table 1. Section 2 presents the proposed probabilistic crowdsourcing model based on GP. The VB inference procedure is described in Section 3. The process to classify new samples (including the case when there are test annotations available) is described in Section 4. A comprehensive experimental validation is presented in Section 5. Section 6 concludes the paper and provides some future outlook.

2. Bayesian modeling

Let $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]^T \in \mathbb{R}^{N\times D}$ be a training set of N D-dimensional samples, with unknown labels $\mathbf{z} = (z_1, \dots, z_N)^T \in \{0,1\}^N$. Let us assume there are R different annotators. Let $R_n \subseteq \{1, \dots, R\}$ denote the subset of annotators who labeled the nth sample, and $N_r \subseteq \{1, \dots, N\}$ the subset of samples labeled by the rth annotator. Finally, $\mathbf{Y} = \{y_n^r \in \{0,1\} \mid n = 1, \dots, N;\ r \in R_n\}$ is the set of labels provided by the R annotators.

Gaussian Processes (GP) model the relationship between samples $\mathbf{X}$ and the corresponding unknown true labels $\mathbf{z}$ in two steps. First, a set of latent variables $\mathbf{f} = [f_1, \dots, f_N]^T$ following a joint Gaussian distribution $\mathrm{p}(\mathbf{f}|\mathbf{\Omega}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K})$ is introduced. The kernel matrix $\mathbf{K} = [k(\mathbf{x}_n, \mathbf{x}_m|\mathbf{\Omega})]_{nm}$ is computed with the kernel function k, which defines an inner product in a (possibly infinite-dimensional) transformed space [1, Chapter 6]. Intuitively, the correlation between each pair of entries of $\mathbf{f}$ is calculated in a transformed space of the original feature space, which allows GP to estimate non-linear decision boundaries. In this work we use the well-known squared exponential (SE) kernel $k(\mathbf{x}_n, \mathbf{x}_m) = \gamma \cdot \exp(-||\mathbf{x}_n - \mathbf{x}_m||^2/(2l^2))$, although other kernels could be used. The kernel hyperparameters $\mathbf{\Omega} = \{\gamma, l\}$ are called *variance* and *length-scale*, respectively.

The second step is to relate the latent variables $\mathbf{f}$ to the unknown true labels $\mathbf{z}$ using a product of Bernoulli distributions:

$$\mathrm{p}(\mathbf{z}|\mathbf{f}) = \prod_{n=1}^{N} \sigma(f_n)^{z_n}(1-\sigma(f_n))^{1-z_n}$$
$$= \prod_{n=1}^{N} \left(\frac{1}{1+e^{-f_n}}\right)^{z_n}\left(\frac{e^{-f_n}}{1+e^{-f_n}}\right)^{1-z_n}, \quad (1)$$

where the sigmoid function σ maps $\mathbb{R}$ into the interval (0,1). In other words, the sigmoid function assigns the probability of belonging to a class depending on the value of the real number f_n. When $\mathbf{x}_n$ belongs to class 1 ($z_n = 1$), only the first factor is considered and a large positive value is expected for f_n. When $\mathbf{x}_n$ belongs to class 0 ($z_n = 0$), only the second factor appears and a large negative value is expected for f_n. Notice that, although a realization of a GP is a continuous real function on the feature space, the sigmoid function transforms it into a Bernoulli parameter. This is a natural generalization of logistic regression [30]. While logistic regression uses a linear combination of the components of $\mathbf{x}_n$, with linear weights to be estimated, GP uses a linear combination of features in a transformed domain (this transformed domain depends on the kernel used) and denotes by f_n the corresponding linear combination. Moreover, the sigmoid function is an infinitely differentiable function, which allows VB to infer the posterior distribution of the latent variable $\mathbf{f}$.

The distributions $\mathrm{p}(\mathbf{f}|\mathbf{\Omega})$ and $\mathrm{p}(\mathbf{z}|\mathbf{f})$ define a standard GP classifier. Now we need to include the crowdsourcing labelling process in our model. Each annotator r is described by their *sensitivity* $\alpha_r := \mathrm{p}(y^r = 1|z = 1)$ and *specificity* $\beta_r := \mathrm{p}(y^r = 0|z = 0)$. Intuitively, α_r and β_r represent the reliability of the rth annotator when labeling samples of class $\mathcal{C}_1$ and $\mathcal{C}_0$, respectively. This model is the same as in [9,12]. Assuming independence between annota-

Table 1
A comprehensive glossary of all the symbols used in this work.

Symbol	Description
$\mathcal{C}_0$ and $\mathcal{C}_1$	Classes 0 and 1, respectively.
N	Number of samples in the training set.
D	Dimension of the feature space.
R	Number of annotators who provided crowdsourcing labels.
$R_n \subseteq \{1, \ldots, R\}$	Subset of annotators who labeled the nth sample.
$N_r \subseteq \{1, \ldots, N\}$	Subset of samples labeled by the rth annotator.
$\mathbf{X} \in \mathbb{R}^{N \times D}$	Matrix containing all the samples in the training set.
$\mathbf{x}_n \in \mathbb{R}^D$	nth sample of the training set.
$\mathbf{x}_* \in \mathbb{R}^D$	New sample whose class is predicted by the proposed method.
$\mathbf{Y}$	Set of annotations provided by the R annotators.
$y_n^r \in \{0, 1\}$	Label provided by the rth annotator for the nth sample.
$\mathbf{y}_* = \{y_*^r : r \in R_*\}$	Annotations provided for the new sample $\mathbf{x}_*$.
$\mathbf{z} \in \{0, 1\}^N$	Underlying real labels for the training set instances.
$z_n \in \{0, 1\}$	Underlying real label for the nth sample.
$z_* \in \{0, 1\}$	Underlying real label for $\mathbf{x}_*$.
$\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_R]$	Sensitivity of each annotator.
$\boldsymbol{\beta} = [\beta_1, \ldots, \beta_R]$	Specificity of each annotator.
a_0^α, b_0^α, a_0^β and b_0^β	Hyperparameters for α and β. (Default: All of them equal to 1.)
$\mathbf{f} = [f_1, \ldots, f_N]^T$	GP modeling the relationship between $\mathbf{X}$ and $\mathbf{z}$.
$f_* \in \mathbb{R}$	GP modeling the relationship between $\mathbf{x}_*$ and z_*.
$k(\cdot, \cdot\|\boldsymbol{\Omega})$	Kernel function depending on a set of parameters $\boldsymbol{\Omega}$.
$\mathbf{K} \in \mathbb{R}^{N \times N}$	Covariance matrix of the prior distribution of $\mathbf{f}$.
$\mathbf{h} \in \mathbb{R}^N$	Vector of prior covariances between f_* and $f_1, \ldots, f_n$.
$c \in \mathbb{R}^+$	Prior variance of f_*.
$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^T$	Variational parameters to be estimated.
$\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$	Diagonal matrix calculated from the variational parameters $\boldsymbol{\xi}$.
m_* and s_*^2	Mean and variance of the approximated posterior distribution of f_*.
$\boldsymbol{\mu}_\mathbf{f}$ and $\boldsymbol{\Sigma}_\mathbf{f}$	Mean and covariance matrix of the posterior distribution of $\mathbf{f}$.
$\delta \in [0, 1]$	Classification threshold.
$\boldsymbol{\Theta} = \{\mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}\}$	Subset of unknown variables of the model.
$\bar{\boldsymbol{\Theta}} = \{\mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}\}$	Set of the unknown variables of the model.
$\bar{\boldsymbol{\Theta}}_\theta = \bar{\boldsymbol{\Theta}} \setminus \theta$	Set $\bar{\boldsymbol{\Theta}}$ minus the element $\theta \in \bar{\boldsymbol{\Theta}}$.
$\sigma(\cdot)$ and $\psi(\cdot)$	Sigmoid and Digamma functions, respectively.
$\mathrm{KL}(\cdot\|\|\cdot)$	Kullback–Leibler divergence.
$\mathrm{p}(\cdot)$ and $\mathrm{q}(\cdot)$	Probability distributions: Assumed known (p) and approximated (q).
$\mathbf{0}$ and $\mathbf{1/2}$	Vector with all the components equal to 0 and 1/2, respectively.

tors, we have the following product of Bernoulli distributions

$$\mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{r=1}^{R} \prod_{n \in N_r} \left[\alpha_r^{y_n^r} (1-\alpha_r)^{1-y_n^r} \right]^{z_n} \left[(1-\beta_r)^{y_n^r} \beta_r^{1-y_n^r} \right]^{1-z_n}, \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_R)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_R)$. Some observations are required at this point. First, this sensitivity-specificity model allows for scenarios where annotators might be non-experts. Second, *spammer* (resp. *adversarial*) annotators are those with α_r and β_r values close to (resp. much lower than) 0.5. Third, notice that exchanging the role of z_n and $1 - z_n$ in Eq. (2) corresponds to exactly the same model but with sensitivities $1 - \beta_r$ and specificities $1 - \alpha_r$. This means that, changing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ accordingly, a certain set of underlying true training labels and their opposite are equally plausible. In Section 3 we provide an initialization of our algorithm that accounts for this ambiguity.

In this work, sensitivities and specificities are treated as stochastic variables, and Beta distributions are used as hyperpriors. This treatment allows us to introduce prior knowledge about these parameters and weigh more certain configurations of them, yielding more accurate results and a better uncertainty quantification of the model. The independence between annotators yields $\mathrm{p}(\boldsymbol{\alpha}) = \prod_{r=1}^{R} \mathrm{Beta}(\alpha_r|a_0^\alpha, b_0^\alpha)$ and $\mathrm{p}(\boldsymbol{\beta}) = \prod_{r=1}^{R} \mathrm{Beta}(\beta_r|a_0^\beta, b_0^\beta)$,where we have removed the dependency on the parameters for simplicity. Recall that $\mathrm{Beta}(\omega|a, b) \propto \omega^{a-1}(1-\omega)^{b-1}$ with mean $<\omega> = a/(a+b)$. During inference, the following expectations will be required (see [1, Exercise 2.11])

$$<\log\omega> = \psi(a) - \psi(a+b), \quad <\log(1-\omega)> = \psi(b) - \psi(a+b), \tag{3}$$

where ψ denotes the digamma function. The parameters a and b can be set to introduce prior knowledge about the expected values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and our confidence on them. When no prior knowledge is available, $a = b = 1$ produces uniform distributions. For instance, these hyper-priors on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are useful to deal with annotators who only provide labels for samples in one of the classes (see [2] for more details about the so-called *"black swan paradox"*).

The proposed joint probabilistic model for the crowdsourcing problem is

$$\begin{aligned} \mathrm{p}(\mathbf{Y}, \mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}) &= \mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta}) \\ &= \mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})\mathrm{p}(\mathbf{z}|\mathbf{f})\mathrm{p}(\mathbf{f}|\boldsymbol{\Omega})\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\beta})\mathrm{p}(\boldsymbol{\Omega}), \end{aligned} \tag{4}$$

where $\boldsymbol{\Theta} = \{\mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}\}$, and $\mathrm{p}(\boldsymbol{\Omega})$ is a flat prior on the kernel parameters $\boldsymbol{\Omega}$. The probabilistic graphical model is depicted in Fig. 1. Yellow nodes correspond to observed variables, namely, the set of features $\mathbf{X}$ and the labels provided by the annotators $\mathbf{Y}$ (discrete). The unknown variables, to be inferred during training, are represented using blue nodes, namely, the real labels $\mathbf{z}$ (discrete), the GP latent variable $\mathbf{f}$ (continuous), the GP hyper-parameters $\boldsymbol{\Omega}$ (continuous), and sensitivity and specificity $\boldsymbol{\alpha}$**,** $\boldsymbol{\beta}$ (continuous).

Now that we have the full probabilistic modeling of our problem, let us briefly describe and explain in words its capabilities and limitations. First we utilize a GP (a prior on the set of functions defined over the feature space) which in combination with the sigmoid function is used to describe the real underlying classifier. Since we do not have access to the output of this classifier, the probability distributions of the labels provided by each annotator given the output of the true classifier is modeled using the sensitivity (when the true label is one) and specificity (when the true label is zero) parameters. These numbers quantify how close each

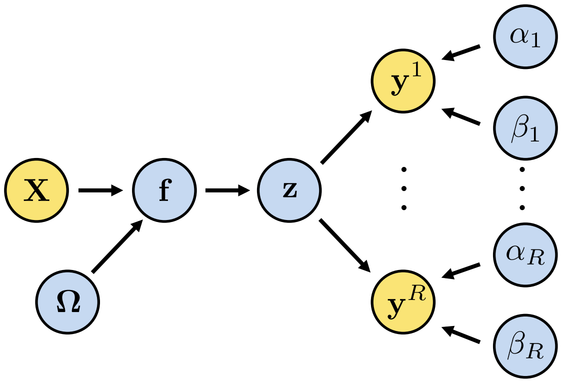

Fig. 1. Probabilistic graphical representation for the proposed model. Yellow nodes denote observed variables, and blue nodes unknown variables (to be inferred during training). $\mathbf{Y} = \{\mathbf{y}^1, \ldots, \mathbf{y}^R\}$ and $\mathbf{z}$ are discrete variables, whereas the rest are continuous. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

annotator's behavior is to the true classifier. Finally, any additional information on each annotator's behavior can be included as prior information on the two aforementioned parameters. In summary, we are using a sound and robust to outliers probabilistic modeling of our crowdsourcing problem.

One of the main limitations of the proposed framework is that the only supervised source of information consists in the labels provided by annotators, lacking some mechanism to introduce additional supervised knowledge which may prevail over the annotators. For instance, there may be some instances in the training set for which we know the real label instead of just (noisy) annotations, in which case we would rather rely on this more accurate knowledge. Second, a simple model has been considered for the annotators, based solely on their sensitivity and specificity. More complex (in particular, feature-dependent) behaviors could happen in real-world problems. For instance, there might be annotators who are much more skilled when labelling instances coming from a certain region of the feature space (because they have specialized in that type of instances), but are not that reliable in other regions. Third, there is the implicit assumption that all the annotators are not spammers. Notice that, for the scenario where all the annotators provide random labels for all the instances, there is no information to be able to infer the true decision boundaries, in which case the proposed method cannot train an accurate classifier.

Having explained the model, let us now see how inference is carried out, what problems will be found when estimating the posterior distribution of all the unknowns given the labels provided by the annotators, and how variational inference can be used to solve all of them.

3. Variational Bayes inference

In Bayesian inference, the main goal is to find the posterior distribution $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y}) = \mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta})/\mathrm{p}(\mathbf{Y})$. This models our certainty about the values of the different model variables once the annotations $\mathbf{Y}$ are observed, and allows us to make predictions on new samples as well as to assess the reliability of the annotators. However, notice that the marginal

$$\mathrm{p}(\mathbf{Y}) = \sum_{\mathbf{z}} \int_{\mathbf{f}} \int_{\boldsymbol{\alpha}} \int_{\boldsymbol{\beta}} \int_{\boldsymbol{\Omega}} \mathrm{p}(\mathbf{Y}, \mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}) \mathrm{d}\mathbf{f} \mathrm{d}\boldsymbol{\alpha} \mathrm{d}\boldsymbol{\beta} \mathrm{d}\boldsymbol{\Omega} \tag{5}$$

is not tractable, and therefore we resort to the approximated Variational Bayes (VB) inference procedure.

In principle, inference for this model could be also addressed through Markov Chain Monte Carlo (MCMC) methods, which involve sampling from the posterior instead of approximating it with an explicit probability distribution. In fact, MCMC was one of the first approaches for approximate inference in GP [31], and its extension to our model is straightforward from a theoretical viewpoint. However, MCMC methods are computationally expensive. This issue is exacerbated when using GPs, since the large number of latent variables (at least one for each training instance) and the high correlation that may exist between them in the posterior usually requires sophisticated and slow MCMC sampling schemes [32]. Moreover, analytical approximations (such as EP or VB) have obtained excellent results while being significantly faster [33].

In order to approximate the posterior $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})$, VB minimizes the Kullback–Leibler (KL) divergence with respect to a generic probability distribution $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$:

$$\begin{aligned}\mathrm{KL}(\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})||\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})) &= \sum_{\mathbf{z}} \int \mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) \log \frac{\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})}{\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})} \mathrm{d}\boldsymbol{\Theta} \\ &= \sum_{\mathbf{z}} \int \mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) \log \frac{\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})}{\mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta})} \mathrm{d}\boldsymbol{\Theta} \\ &\quad + \log \mathrm{p}(\mathbf{Y}).\end{aligned}$$

The KL divergence between two distributions is always non negative, and is zero if and only if they coincide. Therefore, the optimal distribution $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$ in the sense of KL divergence minimization is unique and equals the exact $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})$. Interestingly, notice that we do not need to know the real posterior $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})$ to minimize the KL divergence on $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$: since $\log \mathrm{p}(\mathbf{Y})$ does not depend on $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$, only the joint distribution in Eq. (4) is required.

However, the sigmoids in $\mathrm{p}(\mathbf{z}|\mathbf{f})$ (recall Eq. (1)) prevents us from directly evaluating the KL divergence, since their expectation over a Gaussian cannot be obtained in closed-form. To overcome this problem, a variational lower bound for the sigmoid is used [1, Section 10.6]. Namely, for any $\xi > 0$, we have $\sigma(f) = (1 + \exp(-f))^{-1} \geq \sigma(\xi) \exp\left((f - \xi)/2 - \lambda(\xi)(f^2 - \xi^2)\right)$, where $\lambda(\xi) = (2\xi)^{-1}(\sigma(\xi) - 1/2)$ [1, Eq. (10.149)]. In our case, this bound yields $\mathrm{p}(\mathbf{z}|\mathbf{f}) \geq \mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi})$, where

$$\mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi}) = \prod_{n=1}^{N} \sigma(\xi_n) \exp\left\{ f_n\left(z_n - \frac{1}{2}\right) - \lambda(\xi_n) f_n^2 + \xi_n^2 \lambda(\xi_n) - \frac{\xi_n}{2} \right\}.$$

Plugging this bound into Eq. (4), we have the following lower bound for the joint distribution

$$\begin{aligned}&\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}, \mathbf{Y}) \geq \mathbf{M}(\mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}, \mathbf{Y}, \boldsymbol{\xi}) \\ &\quad = \mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})\mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi})\mathrm{p}(\mathbf{f}|\boldsymbol{\Omega})\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\beta})\mathrm{p}(\boldsymbol{\Omega}),\end{aligned}$$

which in turn produces

$$\mathrm{KL}(\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})||\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})) \leq \mathrm{KL}(\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})||\mathbf{M}(\mathbf{z}, \boldsymbol{\Theta}, \mathbf{Y}, \boldsymbol{\xi})) + \mathrm{const}. \tag{6}$$

Interestingly, notice that $\mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi})$ is quadratic in $\mathbf{f}$, which allows us to compute the expectation over a Gaussian in closed-form. Therefore, we focus now on minimizing (with respect to $\mathrm{q}(\mathbf{z}, \Theta)$) the analytically tractable right-hand side term in Eq. (6), which enforces the left-hand side term (intractable) to be small too. The price for using this bound is a new set of parameters $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^T$ which need to be estimated.

So far, we have used a generic $\mathrm{q}(\mathbf{z}, \Theta)$ for the approximate posterior distribution. However, VB requires the specification of a particular family, from which the best distribution, in the sense of KL divergence, will be chosen. In this work we use the popular *mean field theory* [1, Section 10.1], which assumes that the approximated distribution factorizes as $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) = \mathrm{q}(\mathbf{z})\mathrm{q}(\mathbf{f})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\boldsymbol{\Omega})$. Let $\bar{\boldsymbol{\Theta}} = \{\mathbf{z}, \boldsymbol{\Theta}\}$ be the set $\boldsymbol{\Theta}$ expanded with the variable $\mathbf{z}$. For $\theta \in \bar{\boldsymbol{\Theta}}$, let us write $\bar{\boldsymbol{\Theta}}_\theta = \bar{\boldsymbol{\Theta}} \backslash \theta$ for the set $\bar{\boldsymbol{\Theta}}$ minus θ, and $\mathrm{q}(\bar{\boldsymbol{\Theta}}_\theta) = \prod_{\eta \in \bar{\boldsymbol{\Theta}}_\theta} \mathrm{q}(\eta)$. Then, for each $\theta \in \bar{\boldsymbol{\Theta}}$, it can be shown that the distribution $\mathrm{q}(\theta)$ that minimizes the KL-divergence is given by (see

[1, Eq. (10.9)] for details)

$$\ln \mathrm{q}(\theta) = < \ln \mathbf{M}(\mathbf{z}, \boldsymbol{\Theta}, \mathbf{Y}, \boldsymbol{\xi}) >_{\mathrm{q}(\bar{\boldsymbol{\Theta}}_\theta)} + \mathrm{const}. \tag{7}$$

Alternating the estimation of $\mathrm{q}(\mathbf{z})$, $\mathrm{q}(\mathbf{f})$, $\mathrm{q}(\boldsymbol{\alpha})$, $\mathrm{q}(\boldsymbol{\beta})$ and $\mathrm{q}(\boldsymbol{\Omega})$ leads to an iterative algorithm where the KL divergence decreases after each iteration. Since it is always a non-negative number, the convergence is ensured.

To calculate $\mathrm{q}(\mathbf{z})$, we deduce from Eq. (7) that it factorizes as $\mathrm{q}(\mathbf{z}) = \prod_{n=1}^{N} \mathrm{q}(z_n)$. Thus, we can compute each $\mathrm{q}(z_n)$ separately. Since z_n only takes two values, we have

$$\mathrm{q}(z_n = 0) \propto \prod_{r \in R_n} \exp\{y_n^r < \log(1-\beta_r) > + (1-y_n^r) < \log \beta_r >\}, \tag{8}$$

$$\mathrm{q}(z_n = 1) \propto \exp(<f_n>) \prod_{r \in R_n} \exp\{y_n^r < \log \alpha_r > + (1-y_n^r) < \log(1-\alpha_r) >\}.$$

For $\mathrm{q}(\mathbf{f})$ we observe that $< \ln \mathbf{M}(\mathbf{z}, \boldsymbol{\Theta}, \mathbf{Y}, \boldsymbol{\xi}) >_{\mathrm{q}(\bar{\boldsymbol{\Theta}}_\mathbf{f})}$ cannot be calculated. To avoid this problem, we assume that $\mathrm{q}(\boldsymbol{\Omega})$ is a degenerate distribution. Then, $< \ln \mathbf{M}(\mathbf{z}, \boldsymbol{\Theta}, \mathbf{Y}, \boldsymbol{\xi}) >_{\mathrm{q}(\bar{\boldsymbol{\Theta}}_\mathbf{f})}$ becomes a quadratic function of $\mathbf{f}$ and, therefore, the approximate posterior $\mathrm{q}(\mathbf{f})$ is a Gaussian $\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_\mathbf{f})$. Mean and covariance are calculated by taking first and second order derivatives of $\log \mathrm{q}(\mathbf{f})$ to obtain:

$$\boldsymbol{\mu}_\mathbf{f} = \boldsymbol{\Sigma}_\mathbf{f}(<\mathbf{z}> - (\mathbf{1/2})), \qquad \boldsymbol{\Sigma}_\mathbf{f} = \left(\mathbf{K}^{-1} + 2\boldsymbol{\Lambda}\right)^{-1}, \tag{9}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda(\xi_1), \ldots, \lambda(\xi_N))$, and $<\mathbf{z}> = (\mathrm{q}(z_1 = 1), \ldots, \mathrm{q}(z_n = 1))^T$.

Since $\mathrm{q}(\boldsymbol{\Omega})$ is a degenerate distribution, we only need the value of $\boldsymbol{\Omega}$ where $\mathrm{q}(\boldsymbol{\Omega})$ is not zero. For that, we minimize the following objective function

$$\mathcal{L}(\boldsymbol{\Omega}) = \ln|\mathbf{K} + (2\boldsymbol{\Lambda})^{-1}| + \mathbf{u}^T(\mathbf{K} + (2\boldsymbol{\Lambda})^{-1})^{-1}\mathbf{u}, \tag{10}$$

where $\mathbf{u} = (1/2) \cdot \boldsymbol{\Lambda}^{-1}(<\mathbf{z}> - (\mathbf{1/2}))$. Recall also that $\mathbf{K}$ depends on $\boldsymbol{\Omega}$.

To calculate $\mathrm{q}(\boldsymbol{\alpha})$ and $\mathrm{q}(\boldsymbol{\beta})$, we deduce from Eq. (7) that both factorize as $\mathrm{q}(\boldsymbol{\alpha}) = \prod_{r=1}^{R} \mathrm{q}(\alpha_r)$ and $\mathrm{q}(\boldsymbol{\beta}) = \prod_{r=1}^{R} \mathrm{q}(\beta_r)$. Then we can calculate each $\mathrm{q}(\alpha_r)$ and $\mathrm{q}(\beta_r)$ separately. From Eq. (7) we obtain the following Beta distributions:

$$\mathrm{q}(\alpha_r) = \mathrm{Beta}\left[\alpha_r \middle| a_0^\alpha + \sum_{n \in N_r} <z_n> y_n^r, \ \ b_0^\alpha + \sum_{n \in N_r} <z_n>(1-y_n^r)\right] \tag{11}$$

$$\mathrm{q}(\beta_r) = \mathrm{Beta}\left[\beta_r \middle| a_0^\beta + \sum_{n \in N_r}(1 - <z_n>)(1-y_n^r), \ b_0^\beta + \sum_{n \in N_r}(1 - <z_n>)y_n^r\right]. \tag{12}$$

For $\boldsymbol{\xi}$ we maximize $< \ln \mathbf{M}(\boldsymbol{\Theta}, \mathbf{Y}, \mathbf{X}, \boldsymbol{\xi}) >_{\mathrm{q}(\bar{\boldsymbol{\Theta}})}$ w.r.t. each ξ_n, which yields

$$\xi_n = \sqrt{<f_n>^2 + \boldsymbol{\Sigma}_\mathbf{f}(n,n)}. \tag{13}$$

The whole estimation procedure is summarized in Algorithm 1. Notice that an initial approximated posterior for the true labels $\mathrm{q}^0(\mathbf{z})$ is required. We propose to initialize it with *soft majority voting*, that is, $\mathrm{q}^0(z_n = 1)$ is the proportion of annotators that assign label 1 to the sample $\mathbf{x}_n$. This initialization implicitly assumes that most of the annotators are not adversarial. Otherwise, due to the ambiguity of Eq. (2), we would train a classifier predicting the opposite labels.

4. The predictive distribution

Once the model is trained, we are given a new sample $\mathbf{x}_*$ and we need to predict the probability of each class. In a crowdsourcing problem, we additionally might have access to a set of labels $\mathbf{y}_* = \{y_*^r \in \{0,1\} : r \in R_* \subseteq \{1, \ldots, R\}\}$ provided by some annotators. To the best of our knowledge, this plausible scenario has not been addressed before in the crowdsourcing literature.

Algorithm 1 VGPCR (Variational GP for CRowdsourcing).

```
Require: $\mathbf{X}$, $\mathbf{Y}$, $\boldsymbol{\xi}^0 = \mathbf{1}$, $\mathrm{q}^0(\mathbf{z})$, $k = 0$.
repeat
    Calculate $\boldsymbol{\Omega}^{k+1}$ as the minimizer of Eq. (10) using $\mathrm{q}^k(\mathbf{z})$ and $\boldsymbol{\xi}^k$;
    Update $\mathrm{q}^{k+1}(\mathbf{f})$ with Eq. (9) using $\mathrm{q}^k(\mathbf{z})$, $\boldsymbol{\xi}^k$, and $\boldsymbol{\Omega}^{k+1}$;
    Update $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$ with Eqs. (11) and (12) using $\mathrm{q}^k(\mathbf{z})$ ;
    Update $\mathrm{q}^{k+1}(\mathbf{z})$ with Eq. (8) using $\mathrm{q}^{k+1}(\mathbf{f})$. The expectations are calculated with Eq. (3) using $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$.
    Calculate $\boldsymbol{\xi}^{k+1}$ with Eq. (13) using $\mathrm{q}^{k+1}(\mathbf{f})$;
    $k = k + 1$;
until convergence
output $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$
```

In our model, the prediction can be naturally obtained as the conditional of the hidden label z_* given the observed labels $\mathbf{y}_*$ and $\mathbf{Y}$, that is,

$$\mathrm{p}(z_*|\mathbf{y}_*, \mathbf{Y}) = \sum_{\mathbf{z}} \int \mathrm{p}(z_*, \mathbf{z}, f_*, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}|\mathbf{y}_*, \mathbf{Y}) \mathrm{d}\boldsymbol{\Theta} \mathrm{d}f_* \tag{14}$$

$$\approx \mathrm{const} \cdot \left[\int \mathrm{p}(z_*|f_*) \left(\int \mathrm{p}(f_*|\mathbf{f}) \mathrm{q}(\mathbf{f}) \mathrm{d}\mathbf{f}\right) \mathrm{d}f_*\right] \cdot \left[\int \mathrm{p}(\mathbf{y}_*|z_*, \boldsymbol{\alpha}, \boldsymbol{\beta}) \mathrm{q}(\boldsymbol{\alpha}) \mathrm{q}(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\alpha} \mathrm{d}\boldsymbol{\beta}\right].$$

The GP conditional is $\mathrm{p}(f_*|\mathbf{f}) = \mathcal{N}(f_*|\mathbf{h}^T\mathbf{K}^{-1}\mathbf{f}, c - \mathbf{h}^T\mathbf{K}^{-1}\mathbf{h})$, where $\mathbf{h} = [k(\mathbf{x}_1, \mathbf{x}_*), k(\mathbf{x}_2, \mathbf{x}_*), \ldots, k(\mathbf{x}_N, \mathbf{x}_*)]^T$, $c = k(\mathbf{x}_*, \mathbf{x}_*)$, and $\mathbf{K}$ is the kernel matrix in $\mathbf{X}$. Then, using $\mathrm{q}(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_\mathbf{f})$ (recall Eq. (9)) we obtain $\int \mathrm{p}(f_*|\mathbf{f})\mathrm{q}(\mathbf{f})\mathrm{d}\mathbf{f} = \mathcal{N}(f_*|m_*, s_*^2)$, where $m_* = \mathbf{h}^T\mathbf{K}^{-1}\boldsymbol{\mu}_\mathbf{f}$ and $s_*^2 = c - \mathbf{h}^T\left(\mathbf{K} + (2\boldsymbol{\Lambda})^{-1}\right)^{-1}\mathbf{h}$.

Substituting back in Eq. (14) and using Eq. (4.153) in [1], we obtain the following predictive distribution for z_*:

$$\mathrm{p}(z_* = 1|\mathbf{y}_*, \mathbf{Y}) \propto \sigma(\kappa(s_*^2)m_*) \prod_{r \in R_*} <\alpha_r>_{\mathrm{q}(\alpha_r)}^{y_*^r} (1 - <\alpha_r>_{\mathrm{q}(\alpha_r)})^{1-y_*^r}, \tag{15}$$

$$\mathrm{p}(z_* = 0|\mathbf{y}_*, \mathbf{Y}) \propto \left(1 - \sigma(\kappa(s_*^2)m_*)\right) \prod_{r \in R_*} (1 - <\beta_r>_{\mathrm{q}(\beta_r)})^{y_*^r} <\beta_r>_{\mathrm{q}(\beta_r)}^{1-y_*^r}, \tag{16}$$

where $\kappa(s_*^2) = (1 + \pi s_*^2/8)^{-1/2}$. Notice that this distribution generalizes the case where no information is provided by the annotators, that is $\mathbf{y}_* = \emptyset$. In such a case, the predictive distribution for z_* is the Bernoulli distribution $\mathrm{p}(z_*|\mathbf{Y}) = \left[\sigma(\kappa(s_*^2)m_*)\right]^{z_*}\left[1 - \sigma(\kappa(s_*^2)m_*)\right]^{1-z_*}$. Finally, a threshold δ is used on $\mathrm{p}(z_* = 1|\mathbf{y}_*, \mathbf{Y})$ to assign the new sample $\mathbf{x}_*$ to $\mathcal{C}_1$.

In the next section we will compare our novel crowdsourcing method against current state-of-the-art approaches. We will observe that the proposed method stands out as the most robust approach across a wide range of datasets. In particular, we will see that the proposed VB inference is better suited than EP for GP-based crowdsourcing classifiers. Finally, although the main goal is to illustrate the performance of the proposed method, we will also examine the behavior, strengths, and weaknesses of the other methods it is compared with. To some extent, this provides an up-to-date experimental review of the main crowdsourcing approaches in the literature.

Table 2
The three types of data used in this work.

	Fully synthetic	Semi-synthetic	Fully real
Classif. data set	Synthetic	Real	Real
Annotations	Synthetic	Synthetic	Real
Examples	1D cosine-based	Heart Sonar	Sentence Polarity Music Genre

5. Experimental results

In this section, we provide a comprehensive experimentation that compares the proposed method with several state-of-the-art approaches on three different types of datasets. First, we make use of *fully synthetic* data, where crowdsourcing annotations are synthetically generated for an also synthetic underlying classification dataset. This constitutes a completely controlled framework where we can check the expected behavior of the compared algorithms. Second, we evaluate the methods on *semi-synthetic* data, where the underlying classification dataset comes from a real-world problem but the crowdsourcing annotations are obtained synthetically. This is an interesting and popular hybrid setting in crowdsourcing, where we can keep the influence of the real underlying classification dataset apart from the crowdsourced annotations, which remain under control. Third, we evaluate the methods on *fully real* data, where both features and annotations come from a real problem. This is the most realistic setting for practical applications, although we have no knowledge about the data generation process. Table 2 summarizes the types of data used in this work.

The proposed method is referred to as *VGPCR* (Variational GP for CRowdsourcing). In all the experiments, *VGPCR* is compared against the state-of-the-art GP-based crowdsourcing method in [12] (*Rodrigues*), which utilizes EP as inference procedure. In the comparison we also include the most straightforward manner to apply a GP to the crowdsourcing setting, *GP-MV*, which consists of a standard GP classifier trained with the Majority Voting (MV) labels. The last GP-based method included in the comparison is a GP classifier trained with the true labels (*GP-GOLD*).[2] Notice that, intuitively, *GP-GOLD* and *GP-MV* provide (respectively) upper and lower bounds for the GP-based crowdsourcing methods *Rodrigues* and *VGPCR*. Finally, to obtain a more thorough comparison, we include the methods in [9] (*Raykar*), and [10] (*Yan*), which are based in LR instead of GP (recall the third paragraph in Section 1).

If the annotators provide labels for the test set (that is, some $\mathbf{y}_*$ are available), then our method is referred to as *VGPCR**. As a baseline, we find interesting to compare *VGPCR** with the most straightforward way to predict with the test set annotations, which we refer to as *MV**, and whose predictions are based only on these annotations (no training step is needed). A brief summary of all the algorithms used in the experiments is provided in Table 3.

The predictive performance of the methods is compared using two popular metrics: the area under the ROC curve (AUC), and the overall accuracy (OA), which is calculated for the threshold $\delta = 1/2$. Moreover, in order to compare the computational cost, the CPU time needed to train each method is also provided.

We implemented *VGPCR*$^{(*)}$, *Raykar, Yan*, and GP-classification (necessary for *GP-GOLD* and *GP-MV*) in Matlab©, whereas a Matlab© implementation for *Rodrigues* can be downloaded from his website http://www.fprodrigues.com. All the code and datasets are available at http://decsai.ugr.es/vip/software.html. The experiments were run on the same machine Intel© Xeon© E5-4640 @ 2.40GHz.

5.1. Fully synthetic data

In this section we compare the performance of the methods with a controlled one-dimensional example. Fig. 2a shows the underlying synthetic classification dataset used. The features are uniformly sampled in the interval $[-\pi, \pi]$. The real labels are assigned according to the sign of the cosine function on each sample: class $\mathcal{C}_1$ (resp. class $\mathcal{C}_0$) if the cosine is positive (resp. negative). Then, we simulate $R = 5$ annotators by fixing the values of sensitivity and specificity to $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, \ 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$, respectively. That is, if the true label of the nth sample is $z_n = 1$ (resp. $z_n = 0$), the rth annotator assigns it to class $\mathcal{C}_1$ (resp. $\mathcal{C}_0$) with probability α_r (resp. β_r). In Fig. 2 (b–f) we show the labels assigned by each annotator. As expected from the values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, annotators 1, 2, 3, and 5 make fewer mistakes than annotator 4, who assigns most samples to the opposite class (it has an *adversarial* behavior).

The experiment is repeated 10 times with different training sets of 100 samples (50 of each class). In each realization we also generate a uniformly sampled test set with 200 instances (100 each class). Moreover, test set annotations are also simulated in order to apply the *MV** and *VGPCR** algorithms.

Table 4 shows the predictive performance of all the methods for the 10 realizations. Let us focus first on the five crowdsourcing algorithms that do not use the test set annotations (i.e., the central columns of the table). The results show two clear groups: those based on GP (*GP-MV, Rodrigues, VGPCR*), whose results are competitive with *GP-GOLD*, and those that use LR (*Raykar, Yan*), whose performance is really poor. This is a reasonable behavior if we take into account that LR decision boundaries are hyperplanes, i.e., one point in this 1-D example. This is clearly insufficient to deal with our training dataset, where $\mathcal{C}_0$ has two disconnected parts with $\mathcal{C}_1$ in the middle (recall Fig. 2a)).

Among the LR-based algorithms, we observe that *Yan* performs considerably better than *Raykar*. This means that *Yan*'s feature-dependent model for the annotations is, to some extent, helping to compensate for the insufficient LR model. It is worth noting that the mean result for *Raykar* is hardly above a random guess (around 0.5 of AUC and OA). This LR deficiency is clearly overcome by the GP-based methods, which manage to effectively separate the classes by using a non-linear kernel that allows for more complex decision boundaries (SE kernel in this work, recall Section 2). Among the GP-based methods, the proposed *VGPCR* obtains the best result, followed closely by *Rodrigues*. Notice that *GP-MV* is very close to them in AUC but not in OA, which implies that the threshold $\delta = 1/2$ is not the most appropriate one for class prediction in *GP-MV* (although the classes are well-separated by some other threshold). Nonetheless, this simple 1-D example turns out to be too easy for the GP-based methods, and further differences will be appreciated in subsequent experiments.

It is also interesting to check that, as theoretically intuited, *Rodrigues* and *VGPCR* performances are upper and lower bounded by *GP-GOLD* and *GP-MV*, respectively. Moreover, the differences with *GP-GOLD* are almost insignificant, which means that the crowdsourcing methods are able to extract from the noisy annotations almost the same information as a full GP does from the true labels.

Let us now concentrate on the methods that use test set annotations (*MV** and *VGPCR**). The latter reaches mean AUC and OA of 1.0000 and 99.45% respectively, and manages to totally separate the classes in 8 out of the 10 realizations. These results are better than those obtained by *VGPCR*, which supports the idea that crowdsourcing methods can benefit from the probabilistic inte-

[2] Clearly, *GP-GOLD* can only be trained if there are real labels available for the training set. Of course, this is not common in a real crowdsourcing application (otherwise it could be cast as a standard classification problem). However, in the two real datasets used here the true labels are also provided in order to compare with *GP-GOLD*.

Table 3
An overview of the methods compared in the experiments. From top to bottom, the thick horizontal lines separate non-crowdsourcing methods (*GP-GOLD*), crowdsourcing algorithms that do not use test set annotations, and approaches that do use them.

Algorithm	Description
GP-GOLD	Intuitive upper bound for the GP-based crowdsourcing methods. Trains a GP with the real labels (it is not a crowsourcing algorithm).
GP-MV	Simplest way to apply GP to crowdsourcing (intuitive lower bound). Trains a GP with the majority voting labels.
Rodrigues	State-of-the-art GP-based crowdsourcing method proposed in [12]. EP inference is used.
VGPCR	GP-based crowdsourcing method *proposed here*. Variational inference is used.
Raykar	Based on logistic regression. EM for inference. Proposed in [9]. First probabilistic model for crowdsourcing.
Yan	Based on logistic regression. EM for inference. Proposed in [10]. Annotators parameters depend on the instance they label.
*MV**	Simplest (naive) way to use test set annotations for prediction. It does not need a training step.
*VGPCR**	Straightforward extension to *VGPCR*. *Proposed here*. Probabilistically integrates test set annotations in the prediction.

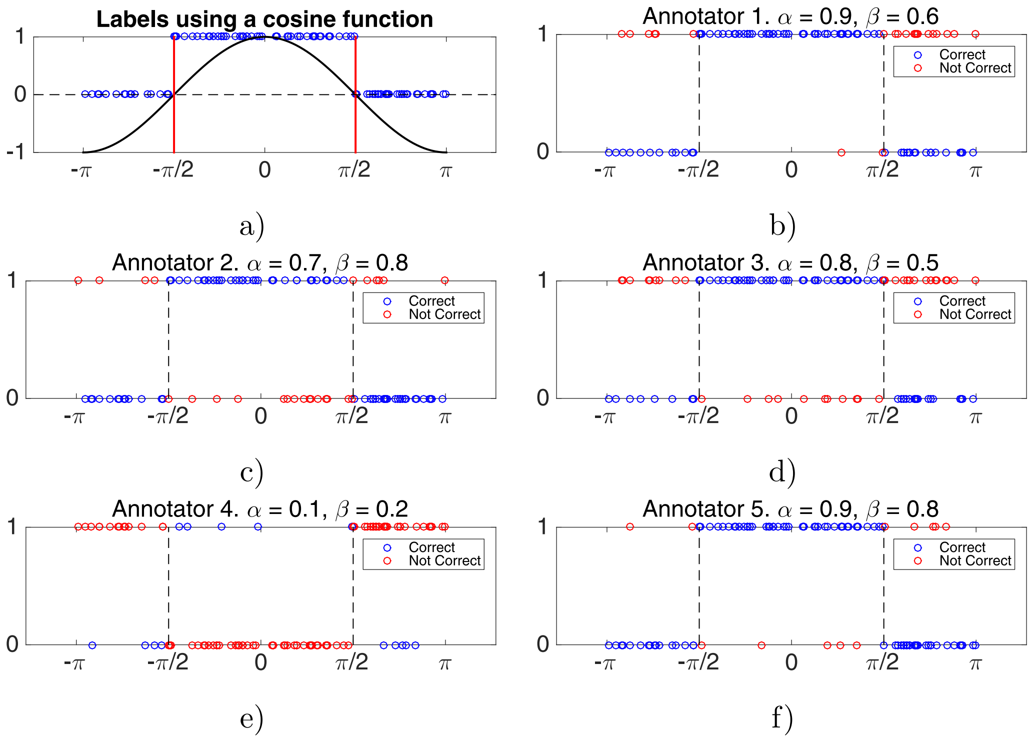

Fig. 2. (a) Original data set labeled using sign of cosine function. (b) - (f) Labels provided by annotators 1,2,3,4 and 5 respectively.

Table 4
Predictive performance of the compared methods for the 10 realizations of the fully synthetic experiment. The best mean performance among the crowdsourcing methods that do not use test set annotations (central columns) is bolded.

	GP-GOLD		GP-MV		Rodrigues		VGPCR		Raykar		Yan		MV*		VGPCR*	
Rea.	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%
1	1.0000	100.00	0.9978	68.00	1.0000	100.00	1.0000	99.50	0.5800	50.00	0.5800	79.00	0.8520	77.00	1.0000	100.00
2	1.0000	98.50	1.0000	84.50	1.0000	98.50	1.0000	98.50	0.4400	50.00	0.5600	57.00	0.8572	78.00	0.9998	99.00
3	1.0000	100.00	1.0000	64.50	1.0000	100	1.0000	99.50	0.4200	50.00	0.7695	75.50	0.7836	76.00	1.0000	100.00
4	0.9998	98.50	0.9987	94.50	1.0000	94.50	1.0000	96.50	0.5000	50.00	0.5000	50.00	0.8357	75.50	1.0000	100.00
5	1.0000	99.00	0.9934	90.50	0.9965	94.50	0.9984	97.00	0.4700	50.00	0.5654	71.00	0.8435	79.00	1.0000	99.50
6	1.0000	100.00	0.9971	91.00	1.0000	100	1.0000	100.00	0.5100	69.00	0.7403	75.00	0.8261	77.00	1.0000	99.00
7	1.0000	99.00	0.9974	96.00	0.9993	98.00	0.9996	98.00	0.5700	76.00	0.6267	71.50	0.8171	76.50	1.0000	99.50
8	0.9990	97.50	0.9928	94.00	0.9972	95.50	0.9983	97.50	0.5500	46.50	0.7435	75.00	0.8436	77.00	0.9997	98.50
9	1.0000	99.50	0.9895	91.00	0.9997	98.00	1.0000	99.50	0.5100	41.50	0.5100	68.00	0.8368	78.50	1.0000	99.50
10	1.0000	98.00	0.9970	78.50	0.9999	98.00	0.9999	96.50	0.4900	50.00	0.7450	73.00	0.8419	79.50	1.0000	99.50
Mean	0.9999	99.00	0.9964	85.25	0.9993	97.70	**0.9996**	**98.25**	0.5040	53.30	0.6340	69.50	0.8337	77.40	1.0000	99.45

gration of test set annotations if available. Moreover, notice that *VGPCR** even outperforms *GP-GOLD* (and this will be also the case in subsequent experiments). In fact, this is expected when the annotations generation process follows the one proposed in the model (like in the synthetic and semi-synthetic experiments) and the sensitivity and specificity parameters ($\boldsymbol{\alpha}, \boldsymbol{\beta}$) are correctly estimated in the training step (as we will check in the following paragraph). In that case, test annotations are a very valuable source of information for *VGPCR**, as they directly depend on the true test label through $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. This has an interesting practical implication in real problems: as long as the crowdsourcing annotation model is representative for the data at hand, it is more useful to collect non-experts opinions for test instances than to collect expert true labels for train ones. Regarding *MV**, its performance is clearly below *VGPCR**. This is reasonable since *MV** does not consider any probabilistic model for the annotations, and thus it is very sensi-

Table 5
Estimated values of sensitivity and specificity for the five annotators in the fully synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		Raykar		Rodrigues		VGPCR	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8910	0.6340	0.8183	0.5433	0.8993	0.6128
2	0.7	0.8	0.6855	0.8410	0.6058	0.7461	0.6828	0.8119
3	0.8	0.5	0.7916	0.4880	0.7607	0.4487	0.8019	0.4837
4	0.1	0.2	0.1362	0.1793	0.2081	0.2724	0.1015	0.1823
5	0.9	0.8	0.8908	0.8168	0.7986	0.6920	0.9042	0.7816

tive to the presence of noise in them (indeed, it performs better as $\boldsymbol{\alpha}, \boldsymbol{\beta} \rightarrow 1$, i.e., when the noise decreases and the annotations themselves become very representative of the underlying true labels).

Finally, Table 5 shows the estimated values of sensitivity and specificity for the models that include them in their formulation (i.e., *Rodrigues*, *VGPCR*, and *Raykar*, since *Yan* uses a more complex feature-dependent model). The proposed *VGPCR* method obtains the most accurate estimations: a maximum absolute difference of 0.0184, whereas it is 0.0410 for *Raykar* and 0.1081 for *Rodrigues* (in next Section 5.2 we will analyze the difficulties of *Rodrigues* to estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$). As stated in the previous paragraph, these reliable estimations of $\boldsymbol{\alpha}/\boldsymbol{\beta}$ in *VGPCR* imply that *VGPCR** greatly benefits from test annotations. Moreover, the estimations of *VGPCR* for annotator 4 are quite accurate, which means that it has been able to recognize its adversarial behavior (*Rodrigues* and *Raykar* also detect it, although less accurately, especially *Rodrigues*). Finally, we stress that the poor performance reported for *Raykar* in Table 4 does not come from a wrong estimation of $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$, but from the underlying LR modeling.

For this simple synthetic experiment, CPU training time is not reported, since all values are very similar (there are only 100 training 1-D instances).

5.2. Semi-synthetic data

In this section we follow an analogous experimental approach as before, but focusing on two more complex semi-synthetic datasets. This allows us to gain additional insight into the behavior of the compared methods. In particular, we observe that the proposed method *VGPCR* (and *VGPCR**) stand out as the most effective and robust crowdsourcing approaches across the two experiments.

5.2.1. Heart dataset

This database, also known as *Heart Disease*, is a popular real classification problem donated by the Cleveland Clinic Foundation to the UCI Machine Learning repository, see http://archive.ics.uci.edu/ml/datasets/Heart+Disease. The goal is to predict the presence or absence (i.e., binary problem) of heart disease in the patient. For that, it contains 13 relevant explanatory variables (features), such as age, resting blood pressure, and maximum heart rate. After discarding 6 instances with missing features, the final dataset contains 297 samples (137 with disease and 160 without it).

With this real underlying classification problem, we simulate $R = 5$ crowdsourcing annotators with the same sensitivity and specificity values as before, i.e., $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$. Notice that the crowdsourcing setting is very appropriate for this medical domain, where different doctors (annotators) may have different opinions (annotations) about the presence/absence of heart disease based on the 13 provided features. The adversarial behavior of annotator 4 represents the meddling of a non-expert annotator who is confusing both classes. We will see that crowdsourcing methods are able to identify this type of undesirable annotator, and take advantage of their opinions in light of their degree of expertise.

Table 6
Results in the *heart* semi-synthetic dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods that do not use test set annotations (central rows) is bolded.

Methods	Test set		Train set		CPU time (s)
	AUC	OA%	AUC	OA%	
GP-GOLD	0.8898	81.91	0.9349	86.25	120.65
GP-MV	0.8633	69.33	0.9133	75.91	46.36
Rodrigues	0.8239	78.09	0.9827	93.46	913.06
VGPCR	**0.8870**	**82.02**	0.9298	86.20	29.21
Raykar	0.8853	80.34	0.9287	86.01	0.54
Yan	0.7396	63.37	0.7944	72.69	625.06
*MV**	0.8211	74.04	0.8350	76.88	0
*VGPCR**	0.9921	95.62	0.9935	96.30	29.21

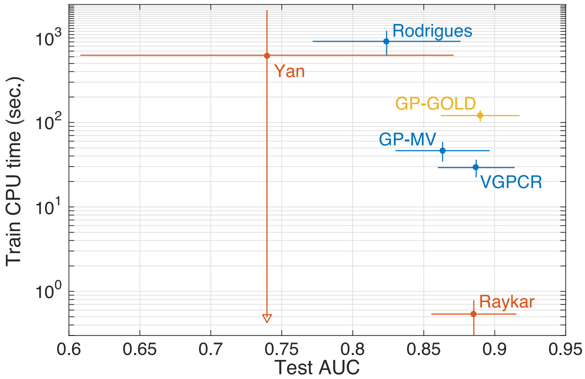

Fig. 3. Trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time) in the *heart* dataset. For each method, the mean plus/minus one standard deviation is shown. The color indicates the family of the algorithm, i.e. yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice also the logarithmic scale in the y-axis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

To average the results over different runs, we consider 10 independent random train/test partitions with 208/89 instances respectively. The results are shown in Table 6 and Fig. 3. The table contains the AUC and OA mean values for both the test and train datasets. Moreover, it shows the mean CPU time needed to train each method. In the figure we focus on two of these quantities, analyzing the trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time). The figure does not include the methods that use test set annotations. Moreover, the figure displays plus/minus one standard deviation of the shown mean quantities. Finally, Table 7 presents the estimated specificity and sensitivity values.

Table 7
Estimated values of sensitivity and specificity for the five annotators in the *heart* semi-synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		Raykar		Rodrigues		VGPCR	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8862	0.6070	0.7058	0.4417	0.8901	0.6065
2	0.7	0.8	0.7052	0.7995	0.5144	0.6310	0.7123	0.8021
3	0.8	0.5	0.8151	0.4772	0.7151	0.3821	0.8177	0.4772
4	0.1	0.2	0.1065	0.1789	0.3808	0.4259	0.1006	0.1790
5	0.9	0.8	0.8807	0.7834	0.6271	0.5554	0.8904	0.7877

We observe that, among the five crowdsourcing methods, the proposed *VGPCR* gets the best generalization (test) performance in both AUC and OA. In fact, in the latter metric it even outperforms *GP-GOLD*, which is trained with the true labels. This means that our method is making the most of the noisy labels that is provided with, reaching the level of its intuitive upper bound. Table 7 also supports that *VGPCR* is able to accurately figure out the model that generates the annotations. The estimated values for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are very close to the true original ones (better than those obtained by *Rodrigues* and similar to *Raykar*'s). In particular, it manages to detect the adversarial behavior of annotator 4. Moreover, it obtains the second shortest CPU train time, 29.21 s.

Regarding its GP-based competitors, the intuitive lower bound *GP-MV* exhibits a worse predictive capacity, as expected. The behavior of *Rodrigues* is, however, more surprising and worth analyzing. *Rodrigues* obtains quite poor test performance, far from *GP-GOLD, VGPCR*, and even its supposed lower bound *GP-MV*. The key is given by its performance in the training set. There, we observe that *Rodrigues* is fitting very well the training instances (e.g. 0.9827 of train AUC), much better than the rest of algorithms (even *GP-GOLD*). This is the so-called *over-fitting* problem, which happens when a machine learning method fits the training data too faithfully, at the expense of its generalization capability. It is also worth pointing that *Rodrigues* is the most computationally heavy method (and with a reduced standard deviation, see Fig. 3). We will see that this inefficiency of the EP inference is recurrent across all experiments, and in Section 5.3.1 we will analyze it in more detail. In the comparison with *VGPCR*, it is also interesting to note that the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ estimates provided by *Rodrigues* are clearly less accurate than those obtained by the proposed method, see Table 7. From a practical viewpoint, this means that *Rodrigues* faces difficulties to identify the annotators reliability, which is a very relevant information for the user. We hypothesize that all this enhancement is due to the two main differences between *Rodrigues* and *VGPCR*: the use of variational inference and the more refined modeling of the annotators. In subsequent experiments, we will further support this idea and analyze some other subtle differences in the modeling of *VGPCR* and *Rodrigues*.

Next, let us analyze the results of the LR-based methods. Interestingly, the simpler model of *Raykar* obtains an excellent test performance, very close to the GP-based *VGPCR* and even *GP-GOLD*. This suggests some latent linear structure in the *heart* dataset. Otherwise, as we saw in the fully synthetic experiment, the LR hyperplanes could not produce as good results as the more complex GP boundaries. In order to confirm this linearity in the underlying *heart* dataset, we trained a standard LR classifier with the true labels (following the same scheme as for the GP-based *GP-GOLD*). As expected, the mean test AUC is 0.8884, almost the same as *GP-GOLD* (recall Table 6). Moreover, *Raykar*'s estimations for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are quite accurate (Table 7), and its computational cost is insignificant (it is the fastest method). Therefore, it could be stated that, in this close-to-linear dataset, the LR-based *Raykar* gets the best trade-off between predictive performance and computational cost. Even better than the proposed GP-based *VGPCR* (see Fig. 3). This supports the common practice in Machine Learning that, when data is simple, a good model does not need to be a complicated one. However, in the remaining experiments we will find more complex datasets where *Raykar* cannot keep up with the level of the proposed *VGPCR*.

On the other hand, in spite of the aforementioned linearity, *Yan* obtains a very poor (the worst) test performance in this dataset. This must be a consequence of its more complex feature-dependent model for the annotations, which makes the convergence at the training step more challenging. This is reflected in the large standard deviations exhibited by *Yan* in Fig. 3, which show that different runs have converged to very different parameters, leading to very heterogeneous results.[3] Moreover, recall that the synthetic generation process used for the annotations does not depend on the features. Therefore, this scenario seems more favorable to *Raykar*, and it will be convenient to compare both methods in the fully real datasets.

Regarding the methods that use test set annotations, the conclusions are the same as in the previous section. Again, *VGPCR** obtains an almost perfect separation between classes, which is mainly caused by the accurate estimation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (recall Table 7). We also observe that the baseline *MV** is not competitive against *VGPCR**, as it is very sensitive to the noisy labels. Recall that *MV** does not need a training step (therefore, its CPU train time is 0).

5.2.2. Sonar dataset

This database, also known as *Sonar, Mines vs Rocks*, is a real classification problem donated by R.P. Gorman and T.J. Sejnowski to the UCI Machine Learning repository, see http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks). The goal is to distinguish between rocks and mines (metal cylinders) by analyzing the sonar signals bounced off these materials. The features are 60 numbers in the range [0,1], where each number represents the energy within a particular frequency band. The dataset includes 208 records, 97 samples correspond to rocks and 111 to mines. For this real underlying classification problem, we simulate $R = 5$ crowdsourcing annotators with the same sensitivity and specificity values as before, i.e., $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$.

To average the results over different runs, we consider 10 independent random train/test partitions with 146/62 instances. As in the *heart* dataset, the results are shown in Table 8 and Fig. 4. The table contains the AUC and OA mean values in both the test and train datasets. Moreover, it shows the mean CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train

[3] In particular, we observed that the surprisingly high mean training CPU time for *Yan* is mainly caused by 2 of the 10 runs, which really struggled to converge. Without them, the mean would be 26.14 seconds, more in accordance with the other LR-based *Raykar*.

Table 8
Results in the *sonar* semi-synthetic dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods that do not use test set annotations (central rows) is bolded.

Methods	Test set		Train set		CPU time (s)
	AUC	OA%	AUC	OA%	
GP-GOLD	0.9043	80.16	0.9901	94.32	98.09
GP-MV	0.7779	60.97	0.8822	71.51	34.05
Rodrigues	0.8574	71.77	0.9843	91.10	153.67
VGPCR	**0.8668**	**74.84**	0.9680	89.59	172.09
Raykar	0.6974	65.65	0.9115	88.08	52.65
Yan	0.6592	57.58	0.7698	75.41	228.72
*MV**	0.8452	77.42	0.8449	77.60	0
*VGPCR**	0.9890	94.19	0.9851	93.15	172.09

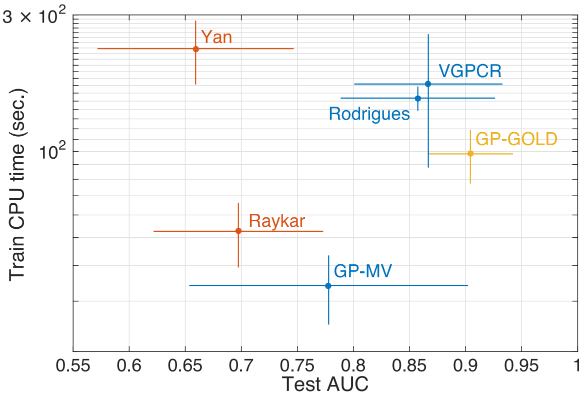

Fig. 4. Trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time) in the *sonar* dataset. For each method, the mean plus/minus one standard deviation is shown. The color indicates the family of the algorithm, i.e. yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice also the logarithmic scale in the y-axis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

time). Finally, Table 9 shows the estimated specificity and sensitivity values.

We observe again that *VGPCR* obtains the best generalization performance among the five crowdsourcing methods that do not use test set annotations (in both test AUC and test OA). Moreover, it clearly obtains the most accurate estimations of specificity and sensitivity: the maximum absolute difference in Table 9 is 0.0266 for *VGPCR*, whereas it is 0.1081 for *Raykar* and 0.3039 for *Rodrigues*. The training CPU time is similar to the one obtained by *Rodrigues* (the only crowdsourcing method that is competitive with it in test performance).

Let us analyze the results for the LR-based methods, which will again shed some light on the internal structure of the underlying classification dataset. As opposed to the *heart* dataset, here the test results for *Raykar* (and *Yan*) are distinctly worse than those for the GP-based methods (specially the more elaborated *VGPCR* and *Rodrigues*, see Fig. 4, which clearly shows that blue points are to the right of red ones in the x-axis). This suggests that this *sonar* database is not as linearly-separable as the one before. Again, this can be confirmed by training a standard LR classifier with the true labels. Indeed, it obtains a mean test AUC value of 0.7546, very far from the more complex decision boundary of *GP-GOLD* (0.9043, see Table 8). Notice also that this LR classifier is an intuitive upper bound for the LR-based crowdsourcing methods (just as *GP-GOLD* is for the GP-based ones). This is in accordance with the test AUC values obtained by *Raykar* and *Yan* (0.6974 and 0.6592, respectively), which are below 0.7546. Regarding the comparison between them, *Yan* is again significantly outperformed by *Raykar*, which is also much faster. The justification is as before: *Yan*'s feature-dependent model is too complex for the simple generation process of the annotations, which follows the simpler model of *Raykar*. This makes the convergence more difficult for *Yan*, whereas *Raykar* logically gets pretty good estimations of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (see Table 9).

The behavior of the GP-based methods is the expected one. Unlike in the *heart* dataset, where it suffered over-fitting, here *Rodrigues* (and also *VGPCR*) exhibit better predictive performance than their intuitive lower bound *GP-MV*. They are also upper bounded by their natural limit *GP-GOLD*. Interestingly, we see that the difference here between *GP-GOLD* and *GP-MV* (in test AUC and OA) is significantly larger than in *heart*. This is connected with the aforementioned non-linearity of the dataset: a close-to-linear boundary can be well identified with low-quality labels, but a complex one needs more accurate data.

The two GP-based methods *VGPCR* and *Rodrigues* present a very similar trade-off between predictive performance and computational cost (Fig. 4). However, the estimations of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are much poorer for *Rodrigues*. In principle this is certainly surprising, because the formulas that define $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in this work (recall Eqs. (11) and (12)) are the same as in *Rodrigues* (see Eqs. (8)–(10) in [12]).[4] There are two explanations for this: 1) the treatment of the latent variable $\mathbf{z}$ (which appears in the formulas for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$), and 2) the modeling of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ themselves. The first one is pretty subtle but very relevant, and refers to the fact that $\mathbf{z}$ is integrated out from the beginning in the model of [12] whereas it is included in our model as a latent variable. That allows us to compute sounder estimates for $\mathbf{z}$, which is the basis of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ update formulas. The second one is clearer, as our posterior distributions over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ account for the uncertainty in the model (whereas the point estimates in [12] do not).

The conclusions for the methods that make use of the test annotations is the same as in the *heart* dataset: *VGPCR** obtains extraordinarily good results thanks to the accurate estimation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, whereas *MV** is not competitive with it because it does not model the noise in the annotations.

5.3. Fully real data

In this section we compare the performance of *VGPCR* and its competitors on two real crowdsourcing datasets. True labels for the training instances are provided by the datasets contributors. This allows us to compare also with *GP-GOLD*. However, no test annotations $\mathbf{y}_*$ are provided, so *VGPCR** and *MV** are not included in this section. The obtained results support that the novel *VGPCR* is also the most competitive approach in these practical applications.

5.3.1. Sentence polarity dataset

The *Sentence Polarity* dataset first was presented by Pang and Lee [34]. It consists of 10,427 sentences extracted from movie reviews in “Rotten Tomatoes” website http://www.rottentomatoes.com/. The goal is to decide whether a sentence corresponds to a “positive” or “negative” review. In Table 10 we show six sentences in the dataset. Preprocessing and feature extraction were carried out by Rodrigues et al. [35], which resulted in feature vectors with 1200 components. The dataset is divided into train and test sets, with 4999 and 5428 samples, respectively. To obtain crowdsourcing labels, the train set was made available in Amazon Mechanical Turk. A total amount of 27,746 labels were obtained from 203 different annotators.

[4] More precisely, recall that we model $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as stochastic variables whereas they are treated as parameters in [12]. Thus, it is the *mean* of the beta distributions in Eqs. (11) and (12) what equals the formulas in [12].

Table 9
Estimated values of sensitivity and specificity for the five annotators in the *sonar* semi-synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		Raykar		Rodrigues		VGPCR	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8889	0.7081	0.6768	0.4374	0.8734	0.6244
2	0.7	0.8	0.6786	0.8548	0.4890	0.6221	0.7027	0.8167
3	0.8	0.5	0.8215	0.5296	0.7003	0.3717	0.8051	0.4761
4	0.1	0.2	0.2054	0.2580	0.3922	0.4850	0.0961	0.2079
5	0.9	0.8	0.8359	0.8253	0.5961	0.5334	0.8815	0.7997

Table 10
Examples of positive and negative samples in *Sentence Polarity* dataset.

Sentence	True label
"An original gem about an obsession with time." "A taut, intelligent psychological drama." "Clever, brutal and strangely soulful movie."	"Positive"
"This is amusing for about three minutes." "The film can depress you about life itself." "The pool drowned me in boredom."	"Negative"

Table 11
Results in the *Sentence Polarity* fully real dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods is bolded.

Methods	Test set		Train set		CPU time (s)
	AUC	OA%	AUC	OA%	
GP-GOLD	0.8037	73.07	0.9130	83.76	3.3089140×10^4
GP-MV	0.7932	72.03	0.8706	79.22	2.9595670×10^4
Rodrigues	0.7815	72.07	0.9415	89.44	1.0685530×10^4
VGPCR	**0.8000**	**72.53**	0.8861	81.32	3.9638080×10^4
Raykar	0.7141	68.22	0.9100	90.68	1.8156210×10^4
Yan	0.7530	69.45	0.8974	84.28	1.4089233×10^5

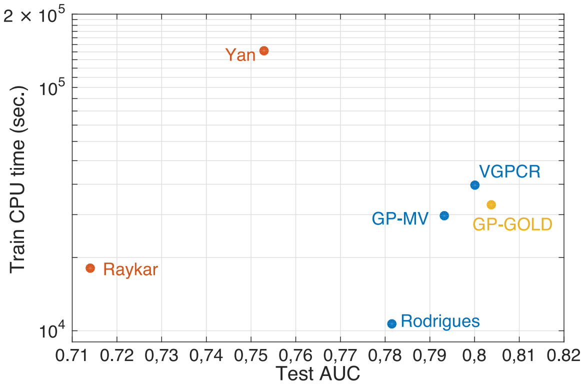

Fig. 5. Trade-off between generalization capability (test AUC) and computational cost (CPU train time) in the *Sentence Polarity* dataset. The color denotes the family of the algorithm: yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice the logarithmic scale in the y-axis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Results are shown in Table 11 and Fig. 5. The table contains the AUC and OA for both test and train datasets. Moreover, it shows the CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train time). Finally, Fig. 6 shows the estimated specificity and sensitivity values.

Again, *VGPCR* is the best crowdsourcing method in terms of predictive performance (0.8 of AUC and 72.53% of OA). These values place the proposed method really close to its natural upper bound *GP-GOLD* (0.8037 of AUC and 73.07% of OA). As expected, it is also lower bounded by *GP-MV*.

It is important here to analyze the behavior of *Rodrigues*. Although it is below its intuitive lower bound *GP-MV* (it is clearly suffering from over-fitting, see its high training performance), its generalization capability is not very far from *VGPCR*, and it seems that it might be the method of choice in certain applications because it is around four (resp. three) times faster than *VGPCR* (resp. *GP-MV*). This low computational cost (the lowest in this dataset) seems certainly surprising, since the EP inference is quite expensive (as both semi-synthetic experiments have shown). The key is that, in this application, the code provided by the authors fixes the kernel hyperparameters from the beginning[5], and they are not estimated during training (which is the most time-consuming step). However, the other two GP-based methods do estimate them. Therefore, the CPU training costs should not be compared. If we fix the kernel hyperparameters of *VGPCR* to its previously estimated values, then its CPU training time falls down to 2.1312×10^3 s (around 5 times less than *Rodrigues*), whereas its predictive performance remains unchanged.

The problem is that, because of the EP inference procedure, estimating the kernel hyperparameters with *Rodrigues* in this dataset (4999 training instances) is computationally prohibitive. Indeed, Fig. 7 shows the experimental CPU time needed to train *Rodrigues* (including the hyperparameters estimation) with increasingly larger subsets of the original set. The rapid growth makes training with $n = 4999$ instances infeasible. Interestingly, the theoretical complexity of each EP iteration is $\mathcal{O}(n^3)$, the same as the variational inference used here. However, EP usually requires many more iterations for convergence, and that makes it computationally heavier in practice.

Regarding the estimation of specificity and sensitivity, Fig. 6 shows very similar estimations for *VGPCR* and *Raykar*, whereas *Rodrigues* deviates from this common tendency. This is in accordance with all the previous experiments, where *Rodrigues* estimations were less reliable. The reasons behind this were analyzed in Section 5.2.2.

Fig. 5 shows a clear separation between GP- and LR-based methods in the x-axis (i.e., the generalization capability). As explained in Section 5.2, this may reveal a non-linear underlying structure in the dataset.

Finally, as opposed to the semi-synthetic datasets, notice that *Yan* significantly outperforms *Raykar* here. This is in accordance with the fact that the annotations generation process does not necessarily imitate *Raykar*'s one in this real dataset, and the feature-dependent model of *Yan* seems to adapt well. However, this is at

[5] Specifically, the length-scale l is fixed to 1.5 and the variance γ to 1.3 (recall Section 2).

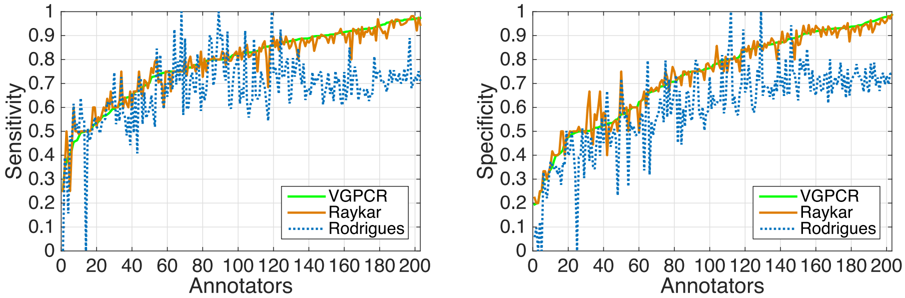

Fig. 6. Sensitivity $\boldsymbol{\alpha}$ (left) and specificity $\boldsymbol{\beta}$ (right) estimations for the 203 annotators in the *Sentence Polarity* dataset. Only those methods that include these parameters in their formulation are shown. For a clearer display, in each figure the annotators are arranged in ascending order of the *VGPCR* estimated value.

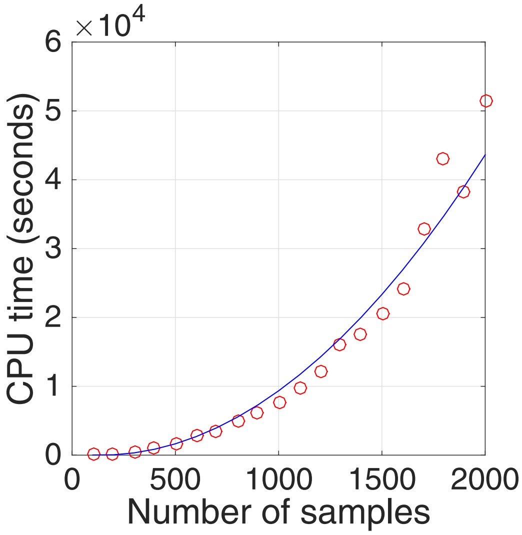

Fig. 7. Mean CPU time in three runs of *Rodrigues*. The x-axis shows the number of training samples.

the expense of a really heavy training step, being the only method (together with Rodrigues) beyond 10^5 seconds of CPU train time.

5.3.2. Music genre dataset

In this experiment we use the Music Genre dataset presented in [36], which consists of 1000 fragments (30 s length) of songs. The goal is to distinguish between 10 music genres: *classical, country, disco, hiphop, jazz, rock, blues, reggae, pop*, and *metal*. We use an *one-vs-all* strategy to address this multi-class classification problem, and the results are averaged over the 10 experiments.

For preprocessing and feature extraction, the authors in [35] used Marsyas music information tool (http://marsyas.info/) to extract 124 features from the original dataset. These features include relevant technical metrics such us means and variances of timbral features, time-domain zero-crossings, spectral centroid, rolloff, flux, and Mel-Frequency Cepstral Coefficients (MFCC).

The dataset contains 100 samples from each genre, which were randomly divided in 70 samples for training and 30 for testing. Crowdsourcing labels were obtained with Amazon Mechanical Turk. Each annotator listened to a subset of fragments and labeled them as one of the ten genres listed above. A total amount of 2945 labels were provided by 44 different annotators.

The results are shown in Table 12 and Fig. 8. The table contains the AUC and OA for both test and train datasets. Moreover, it shows the CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train time). Finally, Fig. 9 shows the estimated specificity and sensitivity values.

Table 12
Results in the *Music Genre* fully real dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods is bolded.

Methods	Test set		Train set		CPU Time (s)
	AUC	OA%	AUC	OA%	
GP-GOLD	0.9426	94.60	0.9713	95.69	3.283342×10^3
GP-MV	0.8865	91.50	0.8809	91.89	2.170080×10^3
Rodrigues	0.8795	85.43	0.9429	92.16	6.520268×10^3
VGPCR	**0.9152**	**92.70**	0.9259	93.70	1.712601×10^3
Raykar	0.8806	90.40	0.9414	95.84	7.201810×10^2
Yan	0.8614	91.90	0.8913	93.96	1.088944×10^3

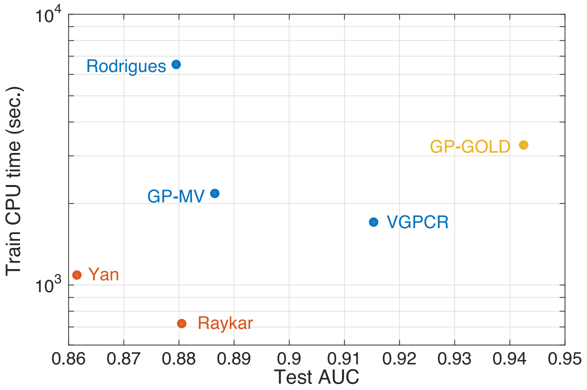

Fig. 8. Trade-off between generalization capability (test AUC) and computational cost (CPU train time) in the *Music Genre* dataset. The color indicates the family of the algorithm: yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice the logarithmic scale in the y-axis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Once more, the novel *VGPCR* exhibits the best generalization capability, keeping a considerable distance with the next one (*GP-MV*). Moreover, *VGPCR* is also the fastest among the GP-based methods. This implies an unbeatable trade-off in Fig. 8. Furthermore, as theoretically expected, its performance lies between that of *GP-GOLD* and *GP-MV*.

As opposed to the previous experiment, *Rodrigues* is now the most computationally expensive method (around three times more than the next one, *GP-MV*). This difference is due to the fact that the kernel hyperparameters are estimated during the training step. Test performance for *Rodrigues* is clearly below *VGPCR*, being only competitive with *GP-MV*. This is due to over-fitting (see its high training performance in comparison with the test one), and the very poor estimation of sensitivity/specificity (see Fig. 9). In turn, as explained in Section 5.2, these follow from a less subtle model-

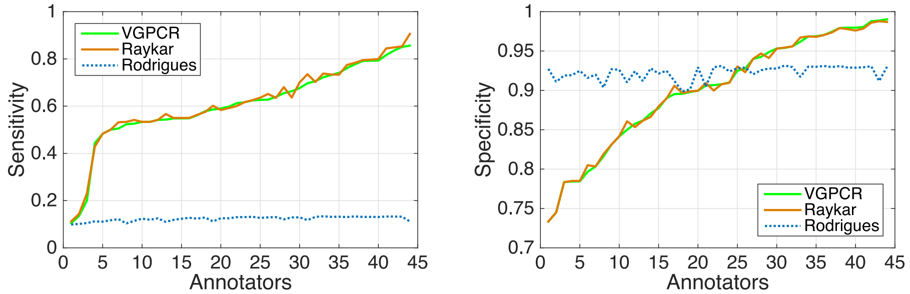

Fig. 9. Sensitivity $\boldsymbol{\alpha}$ (left) and specificity $\boldsymbol{\beta}$ (right) estimations for the 44 annotators in the *Music Genre* dataset. Only those methods that include these parameters in their formulation are shown. For a clearer display, in each figure the annotators are arranged in ascending order of the *VGPCR* estimated value.

ing (marginalization of $\mathbf{z}$, point estimates for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) and the use of a different inference approach.

Once more, Fig. 9 shows similar estimates for *VGPCR* and *Raykar*, whereas *Rodrigues* exhibits a quite bizarre behavior with almost constant estimates. This definitely confirms its difficulties for calculating $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. In this particular case, the problem may come from the unbalanced setting (recall the *one-vs-all* strategy, which implies a 90% − 10% balance between negative and positive classes). In fact, *Rodrigues* is the only method with test OA below 90%, which would be the OA for a naive classifier that assigns every instance to the majority class.

In the comparison between GP- and LR-based methods, the x-axis of Fig. 8 does not show a clear separation in predictive performance. This suggests that linear boundaries may be representative for the classes of this set. Indeed, notice that both families are much better separated in the y-axis. Interestingly, this is precisely connected with the aforementioned underlying linear structure, which allows for a fast convergence of the LR-based methods. The same behavior could be appreciated in the close-to-linear *heart* set, recall Fig. 3.

Regarding the LR-based methods, *Raykar* and *Yan* obtain similar results (the former performs better with respect to AUC and the latter with respect to OA). However, the complex modeling of *Yan* makes it computationally heavier, and thus less competitive in practice.

6. Conclusions

We have introduced a new crowdsourcing classification methodology. As previous approaches, it is based on a Gaussian Process classifier, which allows for the description of complex data. However, a novel Variational Bayes (VB) inference procedure is proposed here (instead of Expectation Propagation, EP). The modeling of the annotators is also refined with respect to previous GP-based methods: the level of expertise is treated as a stochastic variable, and the underlying true training labels $\mathbf{z}$ are not marginalized out from the model. Moreover, the proposed method allows for integrating in the prediction (possibly non-expert) annotations that may have been provided for test instances.

The experimental results have shown that the novel VB-based approach is really competitive and robust across very different types of datasets, ranking always first among its competitors in terms of predictive performance. On the contrary, the EP-based method has suffered over-fitting in three out of the five datasets used. The computational cost of the proposed method is competitive with the rest of the crowdsourcing classifiers (and considerably lower than the EP-based one). Our refined model for the annotators is also reflected in the experiments. Indeed, our sensitivity-specificity estimations are significantly more accurate than those by the EP-based approach. If there are test annotations available (which is not always possible, see the fully real datasets used here), we have seen that the proposed method largely benefits from its probabilistic integration within the model. It would be interesting to study to what extent this generalizes to fully real datasets, where the annotations generation process does not necessarily follow the one proposed in the model. Other lines of future work are i) development of alternative and more accurate feature-dependent crowdsourcing models, ii) a probabilistic multi-class generalization of the proposed model, and iii) extension of GP-based crowdsourcing methods to large-scale datasets.

References

[1] C. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag New York, NJ, USA, 2006.
[2] K. Murphy, Machine Learning: A Probabilistic Perspective, MIT, 2012.
[3] J. Watt, R. Borhani, A. Katsaggelos, Machine Learning Refined: Foundations, Algorithms, and Applications, Cambridge University Press, 2016.
[4] S. Das, S. Datta, B.B. Chaudhuri, Handling data irregularities in classification: foundations, trends, and future challenges, Pattern Recognit. 81 (2018) 674–693.
[5] R. Ekambaram, S. Fefilatyev, M. Shreve, K. Kramer, L.O. Hall, D.B. Goldgof, R. Kasturi, Active cleaning of label noise, Pattern Recognit. 51 (2016) 463–480.
[6] J. Howe, The rise of crowdsourcing, Wired Mag. 14 (6) (2006) 1–4.
[7] J. Zhang, X. Wu, V.S. Sheng, Learning from crowdsourced labeled data: a survey, Artif. Intell. Rev. 46 (4) (2016) 543–576.
[8] A. Dawid, A. Skene, Maximum likelihood estimation of observer error-rates using the EM algorithm, Appl. Stat.-J. Roy. St. 28 (1) (1979) 20.
[9] V. Raykar, S. Yu, L. Zhao, G. Hermosillo Valadez, C. Florin, L. Bogoni, L. Moy, Learning from crowds, J. Mach. Learn. Res. 11 (2010) 1297–1322.
[10] Y. Yan, R. Rosales, G. Fung, M. Schmidt, G. Hermosillo Valadez, L. Bogoni, L. Moy, J. Dy, Modeling annotator expertise: learning when everybody knows a bit of something, in: AISTATS, 2010, pp. 932–939.
[11] Y. Yan, R. Rosales, G. Fung, R. Subramanian, J. Dy, Learning from multiple annotators with varying expertise, Mach. Learn. 95 (3) (2014) 291–327.
[12] F. Rodrigues, F. Pereira, B. Ribeiro, Gaussian process classification and active learning with multiple annotators, in: ICML, 2014, pp. 433–441.
[13] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT, 2006.
[14] P. Ruiz, R. Molina, A. Katsaggelos, Joint data filtering and labeling using gAussian processes and alternating direction method of multipliers, IEEE Trans. Image Process. 25 (7) (2016) 3059–3072.
[15] P. Morales-Álvarez, A. Pérez-Suay, R. Molina, G. Camps-Valls, Remote sensing image classification with large-scale Gaussian processes, IEEE Trans. Geoscie. Remote. 56 (2) (2018) 1103–1114.
[16] F. Rodrigues, M. Lourenco, B. Ribeiro, F. Pereira, Learning supervised topic models for classification and regression from crowds, IEEE Trans. Pattern Anal. 39 (12) (2017) 2409–2422.
[17] S. Wang, S. Chen, T. Chen, X. Shi, Learning with privileged information for multi-label classification, Pattern Recognit. 81 (2018) 60–70.
[18] I. Triguero, C. Vens, Labelling strategies for hierarchical multi-label classification techniques, Pattern Recognit. 56 (2016) 170–183.
[19] J. Duyck, C. Finn, A. Hutcheon, P. Vera, J. Salas, S. Ravela, Sloop: a pattern retrieval engine for individual animal identification, Pattern Recognit. 48 (4) (2015) 1059–1073.
[20] M.V. Giuffrida, F. Chen, H. Scharr, S.A. Tsaftaris, Citizen crowds and experts: observer variability in image-based plant phenotyping, Plant Methods 14 (1) (2018) 12.
[21] Z. Siegel, N. Zhou, S. Zarecor, N. Lee, D. Campbell, C. Andorf, D. Nettleton, C. Lawrence-Dill, B. Ganapathysubramanian, J. Kelly, I. Friedberg, J. Kelly, Crowdsourcing Image Analysis for Plant Phenomics to Generate Ground Truth Data for Machine Learning, Mechanical Engineering Publications, 2018, p. 271.
[22] S. Fritz, L. See, C. Perger, I. McCallum, C. Schill, D. Schepaschenko, M. Duerauer, M. Karner, C. Dresel, J.-C. Laso-Bayas, M. Lesiv, I. Moorthy, C.F. Salk, O. Danylo, T. Sturn, F. Albrecht, L. You, F. Kraxner, M. Obersteiner, A global dataset of crowdsourced land cover and land use reference data, Sci. Data 4 (2017) 170075.

[23] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, N. Navab, Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images, IEEE T. Med. Imaging 35 (5) (2016) 1313–1321.
[24] M. Zevin, S. Coughlin, S. Bahaadini, E. Besler, N. Rohani, S. Allen, et al., Gravity spy: integrating advanced LIGO detector characterization, machine learning, and citizen science, Classical Quant. Grav. 34 (6) (2017) 064003.
[25] F. Rodrigues, F. Pereira, Deep learning from crowds, in: AAAI, 2018, pp. 81–92.
[26] M. Liu, L. Jiang, J. Liu, X. Wang, J. Zhu, S. Liu, Improving learning-from-crowds through expert validation, in: IJCAI, 2017, pp. 2329–2336.
[27] M.V. Gerven, B. Cseke, R. Oostenveld, T. Heskes, Bayesian source localization with the multivariate Laplace prior, in: NIPS, 2009, pp. 1901–1909.
[28] E. Besler, P. Ruiz, R. Molina, A.K. Katsaggelos, Classification of multiple annotator data using variational Gaussian process inference, in: EUSIPCO, 2016, pp. 2025–2029.
[29] P. Ruiz, E. Besler, R. Molina, A. Katsaggelos, Variational Gaussian process for missing label crowdsourcing classification problems, in: MLSP, 2016, pp. 1–6.
[30] J. Bootkrajang, A. Kabán, Learning kernel logistic regression in the presence of class label noise, Pattern Recognit. 47 (11) (2014) 3641–3655.
[31] R.M. Neal, Regression and classification using Gaussian process priors, in: J.M. Bernardo, J.O. Berger, A.P. Dawid, A.F.M. Smith (Eds.), Bayesian Statistic 6, Oxford University Press, 1998, pp. 475–502.
[32] M.K. Titsias, M. Rattray, N.D. Lawrence, Markov Chain Monte Carlo Algorithms for Gaussian Processes, Cambridge University Press, pp. 295–316.
[33] M. Kuss, C.E. Rasmussen, Assessing approximations for Gaussian process classification, in: NIPS, 2006, pp. 699–706.
[34] B. Pang, L. Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, in: Proc. of ACL, 2005, pp. 115–124.
[35] F. Rodrigues, F. Pereira, B. Ribeiro, Learning from multiple annotators: distinguishing good from random labelers, Pattern Recog. Lett. 34 (12) (2013) 1428–1436.
[36] G. Tzanetakis, P. Cook, Musical genre classification of audio signals, IEEE Trans. Speech Audi. Process. 10 (5) (2002) 293–302.

Pablo Ruiz was born in Granada, Spain. He received the M.S. degree in Mathematics, the Master's degree in multimedia technologies and the Ph.D. degree in Information and Communication Technology (ICT) from the University of Granada, in 2008, 2009 and 2015, respectively. He was postdoctoral fellow of the Visual Information Processing group until 2016, at the Department of Computer Science and Artificial Intelligence of the University of Granada. He currently is postdoctoral fellow of the Image and Processing Laboratory (IVPL), which is headed by Prof. Katsaggelos at Northwestern University in Illinois, USA. His research interests focus on the use of Bayesian modeling and inference to solve different inverse problems related with image recovery and machine learning. During his research he has addressed a wide variety of problems such as image restoration, blind image deconvolution, lightfields acquisition, computational tomography, multispectral image classification, active learning, fusion and crowdsourcing.

Pablo Morales-Álvarez received the B.Sc. degree in mathematics and the M.Sc. degrees in mathematical physics and data science from the University of Granada, Granada, Spain, in 2014, 2015, and 2016, respectively, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Artificial Intelligence under the supervision of Prof. R. Molina, funded by a Ph.D. Fellowship from the La Caixa Foundation. His research interests include probabilistic machine learning methods, especially Gaussian processes, and their application to image processing and classification problems.

Rafael Molina received the degree in mathematics (statistics) and the Ph.D. degree in optimal design in linear models from the University of Granada, Granada, Spain, in 1979 and 1983, respectively. He became Professor of Computer Science and Artificial Intelligence at the University of Granada, Granada, Spain, in 2000. He is the former Dean of the Computer Engineering School at the University of Granada (1992–2002) and Head of the Computer Science and Artificial Intelligence department of the University of Granada (2005–2007). His research interest focuses mainly on using Bayesian modeling and inference in problems like image restoration (applications to astronomy and medicine), superresolution of images and video, blind deconvolution, computational photography, source recovery in medicine, compressive sensing, low-rank matrix decomposition, machine learning, active learning, classification, fusion, and crowdsourcing. Prof. Molina serves as an Associate Editor of Applied Signal Processing (2005–2007); the IEEE Transactions on Image Processing (2010–2014); and Progress in Artificial Intelligence (2011–present); and an Area Editor of Digital Signal Processing (2011–present). He is the recipient of an IEEE International Conference on Image Processing Paper Award (2007), an ISPA Best Paper Award (2009), and an European Signal Processing Conference award (2013). He is a coauthor of a paper awarded the runner-up prize at Reception for early-stage researchers at the House of Commons.

Aggelos K. Katsaggelos received the Diploma degree in electrical and mechanical engineering from the Aristotelian University of Thessaloniki, Greece, in 1979, and the M.S. and Ph.D. degrees in Electrical Engineering from the Georgia Institute of Technology, in 1981 and 1985, respectively. In 1985, he joined the Department of Electrical Engineering and Computer Science at Northwestern University, where he is currently a Professor holder of the Joseph Cummings chair. He was previously the holder of the Ameritech Chair of Information Technology and the AT&T chair. He is also a member of the Academic Staff, NorthShore University Health System, an affiliated faculty at the Department of Linguistics and he has an appointment with the Argonne National Laboratory.

He has published extensively in the areas of multimedia signal processing and communications (over 250 journal papers, 600 conference papers and 40 book chapters) and he is the holder of 25 international patents. He is the co-author of Rate-Distortion Based Video Compression (Kluwer, 1997), Super-Resolution for Images and Video (Claypool, 2007), Joint Source-Channel Video Transmission (Claypool, 2007), and Machine Learning Refined (Cambridge University Press, 2016). He has supervised 56 Ph.D. theses so far. Among his many professional activities Prof. Katsaggelos was Editor-in-Chief of the IEEE Signal Processing Magazine (1997–2002), a BOG Member of the IEEE Signal Processing Society (1999–2001), a member of the Publication Board of the IEEE Proceedings (2003–2007), and a Member of the Award Board of the IEEE Signal Processing Society. He is a Fellow of the IEEE (1998), SPIE (2009), EURASIP (2017), and OSA (2018). He is the recipient of the IEEE Third Millennium Medal (2000), the IEEE Signal Processing Society Meritorious Service Award (2001), the IEEE Signal Processing Society Technical Achievement Award (2010), an IEEE Signal Processing Society Best Paper Award (2001), an IEEE ICME Paper Award (2006), an IEEE ICIP Paper Award (2007), an ISPA Paper Award (2009), and a EUSIPCO paper award (2013). He was a Distinguished Lecturer of the IEEE Signal Processing Society (2007–2008).