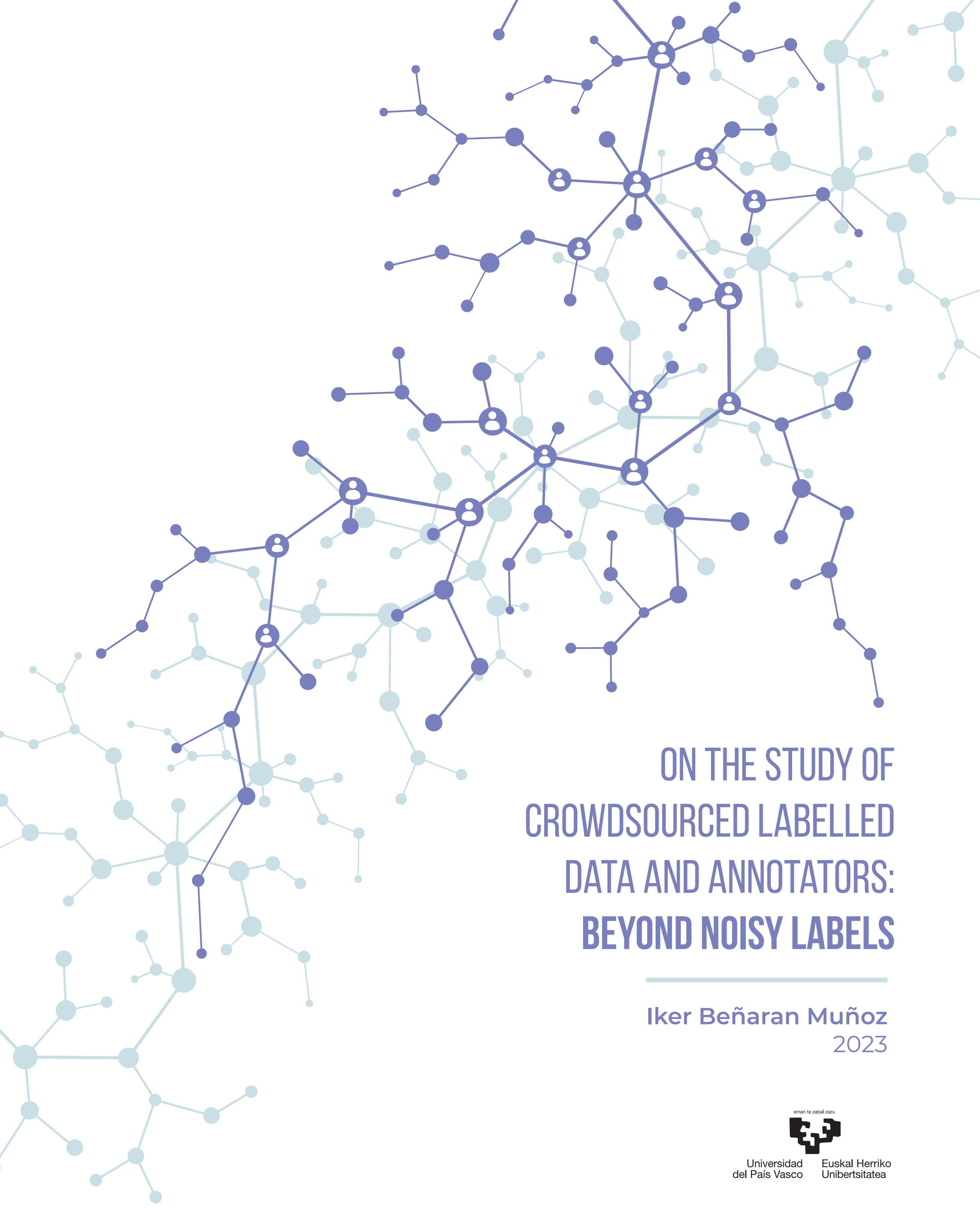


ON THE STUDY OF CROWDSOURCED LABELLED DATA AND ANNOTATORS: **BEYOND NOISY LABELS**

Iker Beñaran Muñoz

2023

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

eman ta zabal zazu

Universidad del País Vasco   Euskal Herriko Unibertsitatea

PhD Thesis in Mathematics & Statistics

On the Study of Crowdsourced Labelled Data and Annotators: Beyond Noisy Labels

by

Iker Beñaran

Supervised by Jerónimo Hernández-González and Aritz Pérez

Donostia - San Sebastián, March 23, 2023

PhD Thesis in Informatics Engineering

On the Study of Crowdsourced Labelled Data and Annotators: Beyond Noisy Labels

by

Iker Beñaran

Supervised by Jerónimo Hernández-González and Aritz Pérez

Dissertation submitted to the University of the Basque Country (UPV/EHU) as partial fulfillment of the requirements for the PhD degree in Informatics Engineering

Donostia - San Sebastián, March 23, 2023

To my family

This research was carried out at the Basque Center for Applied Mathematics (BCAM) within the Machine Learning Group.Iker Beñaran and Aritz Pérez are both supported by the Spanish Ministry MINECO through BCAM Severo Ochoa excellence accreditation SEV-2013-0323, the project TIN2017-82626-R funded by (AEI/FEDER, UE). Iker Beñaran holds a grant no. BES-2016-078095. Aritz Pérez is also supported by the Basque Government through the BERC 2014-2017 and ELKARTEK programs, and by the MINECO through BCAM Severo Ochoa excellence accreditation SVP-2014-068574. Jerónimo Hernández-González is a Serra Húnter Fellow.

Acknowledgments

This thesis would not have been possible without all the help I received from my supervisors, Momo and Aritz. Your scientifical, technical and emotional support has been crucial in so many moments, I thank you from the bottom of my heart. I also want to thank Ekhine Irurozki for allowing me to collaborate as a visiting researcher at Télécom Paris, and Jesús Cerquides IIIA-CSIC for his helpful comments when we discussed part of this work at the IIIA-CSIC.

I also have to thank all the people with whom I have shared office, lunch, beers and conversations, either in the ISG labs, in BCAM or in the IIIA-CSIC labs, especially Amaia for being there through thick and thin, in and out of the workplace.

As they say, the PhD thesis is a long-distance race. There are plenty of people that have given me the strength to keep going on during this race, even if they may not understand a single word written in this work. First of all, my blood family, especially my parents Maria Eugenia and Alvaro, and my sister Oihane. I admire you so much and I truly owe you my life. To my other family RNC, who have been and will always be by my side, *aupa zuek*. To my musical family, especially Eider, Landa, Muña and the whole BRK family. To all the people I have met at the dances and shared the stage (and backstage) with. Also to Eneida, Maritxu and Gal · la, who have been my life partners and essential support in different periods throughout these years. To my current and former flatmates, who have seen me in all kinds of moods. To my psychologists, who have helped me deal with anxiety and many other issues. The list could go on and on, but I had to keep it as short as possible. If our paths have crossed at some point and you feel like you should have been mentioned here, this is also for you.

Contents

1 Introduction

In today's world, the importance of *machine learning (ML)* for managing all the information that is constantly generated is undeniable. ML methods aim to learn from data and build models that can perform tasks as a result of the learning process. To make any ML technique work acceptably, a certain amount of data is required [1], which is not always easy to obtain. Moreover, ensuring the quality of the data gathered is also important to obtain desirable results [2]. A representative example of the importance of having adequate data to avoid obtaining undesirable outcomes is the growing popularity of fairness analysis in ML [3], which aims to audit and mitigate the discriminatory behaviours that ML models learn from the data. In short, bad data may lead to bad models.

One of the most important problems in ML is *supervised classification*, a problem that consists of building a mapping between the descriptive feature space and the set of possible categories or class labels. This mapping is obtained through previous examples of the problem, called training data, in the form of feature-label pairs. Supervised classification is commonly used to deal with prediction and diagnosis problems, in which the outcomes are categorical. As data storage has become cheaper and most electronic devices have been equipped with more and more sensors, events occurring in all parts of the world are continually stored or shared on the Internet. Thus, gathering descriptive data inputs of many different domains is usually not difficult, while categorising the data examples with the real label is often much harder, as it requires some form of external expert judgement. Due to the huge amount of data available, obtaining their corresponding class labels can become an overly costly task. A popular solution to label gathering, as an alternative to domain experts both in supervised classification and in other ML frameworks, is *crowdsourcing*.

1.1 Crowdsourcing

In the last decade, crowdsourcing [4] has become a way of getting large or complex tasks done in a collaborative way. Tasks vary from image labelling to giving opinions about certain products, programming a code or designing a logo, for example. *Requesters* (companies, researchers, etc.), who have a task to be solved, get connected with a set of workers, known as *crowd*, who collaboratively complete the task. That connection is easier now thanks to mobile devices, apps, etc. [5]. Requesters usually divide their original large tasks into smaller and simpler tasks and hand them to the crowd. Workers perform the small tasks assigned to them. Thus, the workload, which would otherwise be unbearable for a single worker, is assumed collaboratively.

A variety of incentives are usually offered for workers to complete their assignments, being economic incentives the most common ones. Online platforms such as *Amazon mTurk*[1], *MicroWorkers*[2] or *Appen*[3] allow requesters to ask crowd workers to complete simple tasks such as filling in surveys or labelling images, for example. In exchange, workers receive a small payment from the requesters. In other cases, motivation is sought through other means such as gamification [6] or the satisfaction of voluntarily contributing to advances in scientific research or society, what is known as *citizen science* [7]. Some examples of citizen science include projects as diverse as the reconstruction of the global surface temperature record back to the decade of the 1780s[4] or the timely assessment of the severity of the damage caused by an earthquake by means of social media pictures [8]. Many more examples of incentives used in crowdsourcing can be found in [9].

In the ML community, crowdsourcing has spread as a way of obtaining large amounts of labelled data [10] at a relatively low cost. Its use has become popular, especially for supervised classification problems as expert labelling tends to be costly in terms of time and money, partly because of the low availability of experts. The huge amount of unsupervised data available makes label gathering even more difficult. In machine learning, the crowd workers, who provide labels for the data, are called *annotators*. The main issue of crowdsourced labelling is that the expertise of the crowd annotators is not guaranteed and their labelling might be misleading. Moreover, the level of expertise among the workers is usually not homogeneous, and so is the quality of the labels they provide. To work around this problem, each example is usually provided to many annotators for labelling. The labels provided by them are commonly used to obtain a consensus or aggregated label, as an estimate of the real class label or ground truth. The consensus label is assumed to be more reliable than each individual annotation and thus the uncertainty associated with the labelling is reduced [11, 12].

[1] https://www.mturk.com/
[2] https://www.microworkers.com/
[3] https://appen.com/crowd-2/
[4] https://www.zooniverse.org/projects/p-teleti/weather-rescue-at-sea

The *learning from crowds* [13] problem consists of learning a classifier, as in standard supervised classification, but using a training set composed of crowd-labelled examples. The difference is that the labels collected from the crowd, which are a type of weak supervision [14], are the only information available about the class variable at training time. The learning from crowds process is illustrated in Figure 1.1. By the nature of crowdsourcing, the learning from crowds techniques are often focused on being cost-effective, that is, to reach the best results counting on the minimum resources. This involves working with non-expert annotators and requesting as few labels as possible.

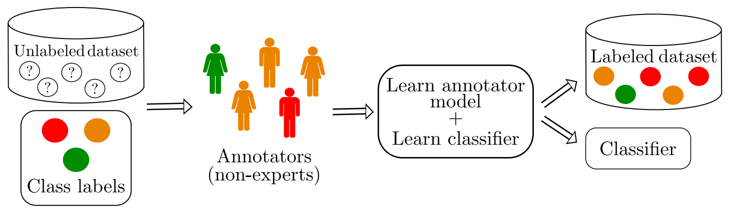


Fig. 1.1: Standard learning from crowds scheme.

Learning from crowds can be divided into two basic tasks. Firstly, *label aggregation* is the process of inferring the consensus label from the inconsistent labels provided by the annotators as an estimate of the ground truth (illustrated in Figure 1.2). Secondly, we learn a predictive model or classifier from the descriptive data and the aggregated labels. There are usually two ways of tackling these two tasks: *sequentially* or *jointly*. Sequential learning approaches focus first on the label aggregation step to obtain high-quality estimates of the ground truth. When the label estimates are obtained, standard supervised learning techniques can be applied using the newly aggregated labels for training the classifier. Joint learning approaches perform the aggregation and the classifier learning stages simultaneously, using learning techniques adapted to deal with crowdsourced labelled data.

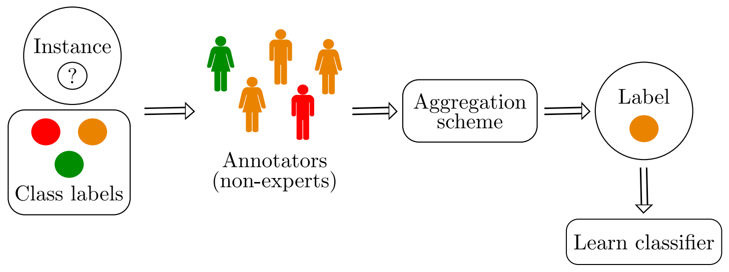


Fig. 1.2: Crowdsourced label aggregation scheme.

To sum up, crowdsourcing is nowadays used extensively for gathering labels for supervised classification due to its availability and cheapness. For this reason, many researchers have proposed diverse approaches to learning from crowds. In this thesis, we provide novel techniques for improving the efficiency of learning from crowds. The aim is to profit from the available information as much as possible, either from the descriptive feature of the instances or from the annotators. In that way, the techniques developed in this thesis show that competitive results can be reached involving fewer annotators and, hence, at a lower cost. In the following section, we will present formally some background concepts before presenting the problems treated in this dissertation.

1.2 Background

Let us formalise the background concepts of this thesis and provide a notation for them. All the symbols presented in this section are described in Table 1.1.

1.2.1 Supervised classification

In this thesis, we deal with supervised classification problems. The objective is to learn a model that classifies as well as possible a data point given in terms of the descriptive feature with a label from a discrete set of class labels.

Formally, let us denote by X the feature that describes our domain of interest, which is a d-dimensional random variable. Instances of that variable, x, take values in the space Ω_X. Let us also denote the categorical class variable C, taking values in the set of labels $\Omega_C = \{1, \ldots, r\}$. A multi-class classification problem is considered, that is, the number of categories is equal or greater than two[1] ($r \geq 2$). Each value $c \in \Omega_C$ is known as *class label*. Each instance is assumed to belong to a single category, a.k.a. *ground truth*. We assume that the random vector (X, C) is distributed according to a probability distribution $p(X, C)$ that is unknown.

In standard supervised classification, we aim to infer, from a training set of n instance-label pairs (x, c_x), a mapping or *classifier* h from the descriptive data space Ω_X to the set of possible class labels Ω_C ($h \colon \Omega_X \to \Omega_C$). It is assumed that the provided class labels c_x are the true labels of their associated instances. The main goal is to obtain a classifier h that generalises well to unseen domain instances. That means that, given a loss function (e.g., the standard $0-1$ loss), h minimises the expectation of the loss function or, in practice, the empirical expected loss on the training set.

When the full information about the real class labels is not provided, the problem is said to provide weak supervision [14]. Many different problems can be grouped under the umbrella of weakly supervised classification. For

[1] Binary classification ($r = 2$) is a particular case of multi-class classification

Symbol	Description
X	Descriptive variable
d	Dimension of the descriptive variable
x	Instance of X
Ω_X	Feature space
C	Class variable
Ω_C	Set of possible class labels (Label space)
r	Number of classes
c	Class label, $c \in \Omega_C$
p	Generative distribution of the problem
n	Total number of instances
c_x	Real label of instance x, $c_x \in \Omega_C$
h	Classifier
$\mathcal{D}$	Set of training instances, $\mathcal{D} \subset \Omega_X$
A	Set of available annotators
a	Annotator from A
m	Number of annotators in A
A_x	Subset of annotators that label instance x, $A_x \subseteq A$
l_x^a	Label provided by annotator a for instance x, $l_x^a \in \Omega_C$
$\mathcal{L}_x$	Labelling for instance x, $\mathcal{L}_x = \{l_x^a\}_{a \in A_x}$
$\mathcal{L}$	Labelling for the whole training set, $\mathcal{L} = \{\mathcal{L}_x\}_{x \in \mathcal{D}}$
$a(\hat{\mathcal{L}})$	Accuracy of the aggregated labelling $\hat{\mathcal{L}}$
ν	Voting function
v	Voting estimate
$\boldsymbol{\alpha}$	Set of parameters of the annotator model
$\Omega_{\boldsymbol{\alpha}}$	Space of parameters of the annotator model
$q(c\|x)$	Estimate of the probability that instance x belongs to class c

Table 1.1: Notation used in this thesis.

example, in the partial labels problem [15] the supervision information of each instance is a set of candidate labels, which always includes the true class label. Another example of weak supervision is *learning from crowds* [13], on which we focus in this thesis.

1.2.2 Learning from crowds

Learning from crowds is a weakly supervised learning problem where true class labels are not given. Instead, several noisy labels are collected for each example from non-expert contributors to face uncertainty.

In learning from crowds, a training set $\mathcal{D}$ with n unlabelled instances is provided. The real class label c_x of the instances $x \in \mathcal{D}$ is unknown, and the only information on supervision available is provided by a crowd of annotators. Let A be the set of available annotators. *Crowd labelling* is the process of getting noisy labels for the instances in the training set from the set A. In this sense, an annotator $a \in A$ can be seen as a classifier which provides labels with a certain amount of noise. We will denote the total number of annotators in A by $m = |A|$. Each instance x is labelled by a subset of annotators $A_x \subseteq A$.

Generally, each annotator $a \in A_x$ provides one label $l_x^a \in \Omega_C$ of their choice. Along this thesis, we refer to this labelling scheme as *full labelling*. Thus, apart from $\mathcal{D}$, the available data for model training in standard learning from crowds is the set of labels for each instance of the dataset, $\mathcal{L} = \{\mathcal{L}_x\}_{x\in\mathcal{D}}$ with $\mathcal{L}_x = \{l_x^a\}_{a\in A_x}$, called *labelling*. Under realistic fair conditions [11], $\mathcal{L}_x$ provides relevant information about the true (unknown) class c_x of instance x. Given $\mathcal{D}$ and $\mathcal{L}$, the goal remains that of supervised classification: to learn a classifier $h\colon \Omega_X \to \Omega_C$ that shows a robust predictive performance on unseen examples.

As aforementioned, learning from crowds can be divided into two subtasks. The first one, *label aggregation*, can be formally defined as the procedure of assigning an estimated class label, which we denote by $\hat{c}_x$, to each instance $x \in \mathcal{D}$. The label assignment is made based on the information at hand: the instances and their collections of labels, x and $\mathcal{L}_x$. The goal is to recover the true labels c_x for each $x \in \mathcal{D}$. The set of aggregated labels for every instance in $\mathcal{D}$ is $\hat{\mathcal{L}} = \{\hat{c}_x : x \in \mathcal{D}\}$. The goal of label aggregation is to find a labelling $\hat{\mathcal{L}}$ that maximises the *aggregation accuracy* (or, more briefly, *accuracy*):

$$a(\hat{\mathcal{L}}) = \frac{1}{n} \sum_{x\in\mathcal{D}} \mathbb{1}(\hat{c}_x = c_x), \tag{1.1}$$

where $\mathbb{1}(cond) = 1$ if $cond = true$ and 0 otherwise.

A simple yet effective approach to label aggregation is the *majority voting (MV)* function, which assigns to each instance x the most frequent label in $\mathcal{L}_x$. Formally, MV function for x can be written as

$$\nu(\mathcal{L}_x) = \arg\max_{c\in\Omega} v(c|\mathcal{L}_x), \tag{1.2}$$

where $v(\cdot|\mathcal{L}_x)$ is the *voting estimate*, which corresponds to the maximum likelihood estimate of the class:

$$v(c|\mathcal{L}_x) = \frac{1}{|\mathcal{L}_x|} \sum_{l\in\mathcal{L}_x} \mathbb{1}(l = c), \tag{1.3}$$

The robustness of MV has been extensively studied [11] and, as long as many annotators take part, it is not required that they have large expertise to obtain an acceptable aggregation accuracy [16].

In order to enhance the label aggregation process, most of the approaches in the state of the art resort to modelling the reliability of the annotators. In that way, the contribution of each labeller in the label aggregation process can be weighted according to their expertise or reliability. Let $\boldsymbol{\alpha}$ be the set of parameters of the annotator model, from the space $\boldsymbol{\Omega}_{\boldsymbol{\alpha}}$. For the sake of simplicity, we may refer to the model itself as $\boldsymbol{\alpha}$ as well. As an example, let us define a simple toy model with parameters $\boldsymbol{\alpha} = \{\alpha_a\}_{a \in A}$, where α_a is the probability that annotator a labels an instance correctly. A way of applying this model to aggregate the crowdsourced labels would be through *weighted voting*, where the vote of each annotator has a weight equal to the model parameter associated with them:

$$v(c|\mathcal{L}_x, \boldsymbol{\alpha}) = \left(\frac{\alpha_a}{|\mathcal{L}_x|} \sum_{a \in A_x} \mathbb{1}(l_x^a = c) \cdot \alpha_a \right). \tag{1.4}$$

This voting estimate can be combined with Eq. 1.2 to obtain a valid label aggregation function. But models used in practice are more complex. Through this dissertation, we use different annotator models. All of them assume that annotators (i) provide labels independently and (ii) tend to provide the correct label with the highest probability among the possible class labels. We consider that annotators are neither adversarial, who provide incorrect labels deliberately, nor colluding, who agree with each other before providing their annotations.

The use of an annotator model introduces a new question: What are the real values of the annotator model parameters? If the real class labels of the data were known, the annotator model could be learned from them. For example, in the previous toy model, we could know the proportion of cases in which each annotator is right or wrong and tune the parameters accordingly. However, in our setting real labels c_x are not available. Alternatively, we could use the estimated class labels aggregated from the labels provided by the annotators to learn the model parameters from them. But note that, in this case, we require label estimates to fit the annotator model ($\hat{\mathcal{L}} \rightarrow \boldsymbol{\alpha}$), and we require a fit of the annotation model to estimate the class labels ($\boldsymbol{\alpha} \rightarrow \hat{\mathcal{L}}$). This mutual requirement naturally leads to the *Expectation-Maximisation* (EM) algorithm [17]. To manage the absence of the true labels and compute estimates for the parameters of the annotator model, many approaches to learning from crowds rely on this approach.

1.2.3 EM algorithm

The EM algorithm is an iterative procedure for finding *maximum likelihood estimates (MLE)* for the parameters of a model given the data. The EM can be used even for data with missing information. In crowd learning scenarios, the EM has used to estimate the maximum likelihood estimates of the reliability parameters $\boldsymbol{\alpha}$ of an annotator model without having the real class labels. We define the likelihood of the data as the probability of the data $\mathcal{D}_c$ given the model parameters $\boldsymbol{\alpha}$, i.e., $Pr(\mathcal{D}_c; \boldsymbol{\alpha}) = \prod_{(x,c)\in\mathcal{D}_c} Pr((x,c); \boldsymbol{\alpha})$. Here we denote $\mathcal{D}_c$ the *complete data*. Additionally, in our setting, the *observed data*, $\mathcal{D}$, has missing information, e.g., the instances have noisy labels given by a set of annotators. In each iteration, the EM maximises the conditional expectation of the likelihood of the missing information given the observed data, using the missing information found in the previous iteration. The EM converges to a local optimum of the marginal likelihood, which is the probability of observing $\mathcal{D}$ given the model parameters: $Pr(\mathcal{D}|\boldsymbol{\alpha})$. In this thesis, we use the EM to deal with the missing information of the class variable and the unknown reliability parameters of the annotators simultaneously.

Let us follow McLachlan and Krishnan [18] formalisation to present the EM procedure to learn the parameters of a model given the incomplete observed data. An EM algorithm starts with an initial guess of the model parameters, $\boldsymbol{\alpha}^{(0)}$. Then, it operates through the interleaved iteration of two steps: (i) Expectation (*E-step*), where the expected value of the unobserved data is computed using the current model fit, and (ii) Maximisation (*M-step*), where new MLE of the model parameters $\boldsymbol{\alpha}$ are obtained given the data completed with the previously computed expected values. In each iteration t, using the previous fit of the model $\boldsymbol{\alpha}^{(t-1)}$, we maximise the conditional expectation of the log-likelihood for the complete data $\mathcal{D}_c$ given the observed data $\mathcal{D}$:

$$Q(\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t-1)}) = E_{\boldsymbol{\alpha}^{(t-1)}}\{\log Pr(D_c; \boldsymbol{\alpha})|\mathcal{D}\}. \tag{1.5}$$

E-step: Compute $Q(\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t-1)})$, for every α.
M-step: Choose $\boldsymbol{\alpha}^{(t)}$ that maximise the conditional expectation of the log-likelihood:

$$\alpha^{(t)} = \arg\max_{\boldsymbol{\alpha}\in\Omega_\alpha} Q(\boldsymbol{\alpha}; \boldsymbol{\alpha}^{(t-1)}). \tag{1.6}$$

The E-step and the M-step are interleaved until convergence. The EM strategy is guaranteed to converge to a local maximum of the likelihood function or, in some exceptional cases, to a saddle point [17], but it is not guaranteed to reach the global maximum. Because of that, the process is usually run many times introducing some random component in the initial guess of the parameters, in order to obtain more than one result. Finally, the model with the highest likelihood among all runs is kept.

In the context of crowd learning, EM allows for fitting an annotator model, with a set of parameters $\boldsymbol{\alpha}$, that typically reflects the reliability of the annotators. The class variable C can be considered latent, and the observed data are $(\mathcal{D}, \mathcal{L})$, that is, the collection of training instances and the collection of crowdsourced labels. The complete data is the collection of all instances with their crowdsourced labels and real class labels: $\mathcal{D}_c = \{(x, \mathcal{L}_x, c_x)|x \in \mathcal{D}\}$. In this case, we have

$$Q(\boldsymbol{\alpha}, \boldsymbol{\alpha}^{(t)}) = E_{\boldsymbol{\alpha}^{(t-1)}}[\log Pr((\mathcal{D}, \mathcal{L}, \boldsymbol{c}))] \tag{1.7}$$
$$= \sum_{\boldsymbol{c}} Pr((\mathcal{D}, \mathcal{L}, \boldsymbol{c}); \boldsymbol{\alpha}^{(t-1)}) \cdot \log Pr((\mathcal{D}, \mathcal{L}, \boldsymbol{c}); \boldsymbol{\alpha}), \tag{1.8}$$

which comes from the discreteness of $\boldsymbol{c}$, where $\boldsymbol{c}$ is the set of all the possible allocations of variable C to each instance, with all $c \in \Omega_C$.

The learning from crowds problem and the concepts presented in this section have been extensively studied in the literature. In the next section, the state-of-the-art key works in relation to crowdsourcing and supervised classification with crowdsourced labels are described.

1.3 Related work

Since its first uses, crowdsourcing has been applied to a variety of fields with very diverse purposes: labelling of data examples [19], text correction [20], text translation [21] and various forms of disease diagnosis [6, 22]. A survey of different applications of crowdsourcing is available in [23]. In this thesis, we are interested in the first application mentioned above, data labelling, which is followed by the learning from crowds problem.

As aforementioned, the approaches to learning from crowds can be roughly divided into two groups. Firstly, *sequential learning* approaches [12, 16, 24, 25, 26, 27, 28] aggregate first the labels provided by the annotators (first stage) and, once the training set can be considered as fully labelled (with the consensus label), a standard supervised learning algorithm is used to learn a classifier (second stage). Secondly, *joint learning* schemes [13, 29, 30, 31] learn the classifier directly from the crowdsourced labels. An extensive review of different learning from crowds techniques, both sequential and joint, can be found in [32].

As for sequential learning methods, probably the most popular one is *majority voting* (MV) [33], which also turns out to be the simplest one: it assigns to each instance the label that most annotators have selected (see Equation 1.2). This strategy is used in many approaches either explicitly or implicitly, and it has been extensively studied, not only in the learning from crowds context [11, 12, 16] but also in other frameworks that have a similar functioning, such as classifier ensembles [34]. As long as many annotators take part, it is not necessary that they are experts for obtaining good results [16]. This method stands out as it does not model the crowd, and it works under the assumption that the contribution of all annotators is equally reliable. Sheng et al. [12] make use of MV in different scenarios including active learning [35], a framework where new labels can be requested in order to reduce uncertainty and to improve the quality of the aggregated labels. They study this aggregation method and its influence in the quality of the aggregated labels. In [16], three variations of MV for the binary case are presented, as well as two aggregation techniques that are based on generating two new (weighted) instances for each example. One of the newly generated instances has a positive label and a weight associated with the number of positive labels received from the crowd, and the other one has a negative label and a weight related to the number of negative labels received from the annotators.

However, crowds are not always homogeneous and the available annotators usually show different levels of expertise [36]. Furthermore, a part of the annotators may be malicious [37]: They could provide misleading information on purpose [38] or act as spammers completing the tasks randomly to get the (monetary) reward without putting enough effort to provide an acceptable answer [39]. Taking that into account, many approaches aim to enhance the voting process by assigning a weight to each annotator's vote depending on their level of expertise or reliability. That is called *weighted voting* [11, 26], and it requires modelling the reliability of the annotators to obtain the corresponding weights. The approach by Karger et al. [26] for binary classification performs voting after computing the weights through a belief propagation-like method. Snow et al. [11] model annotators with a multinomial model similar to naive Bayes and each annotator's vote is weighted by the log-likelihood of the model related to their provided answer. Labellers that are accurate on average obtain positive weights, annotators who label at random have zero influence and adversarial workers obtain negative weights. Weighted voting has also been studied in the framework of classifier ensembles, with results that are applicable in the learning from crowds context as well [40]. There are other aggregation methods that are not based on voting schemes, such as the one by Zhang et al. [28], which combines two clustering processes, each one related to a different type of features extracted from the instances and their associated labels. A class label is assigned to the instances of each cluster during the first process. The second process is used to correct the results from the previous one.

Many learning from crowds methods were derived from the EM algorithm [17] (see Section 1.2.3). This strategy has been used to estimate the (probabilistic) labelling of the training instances (usually previous to learning, in sequential approaches) [24, 25, 41, 42, 43], and also to directly learn a classifier (in joint learning approaches) [13, 30, 31]. It has been combined with spectral methods [42] and deep learning [30], among others. Usually, in these methods, there is not an explicit weighted voting step.

The key work by Dawid and Skene [24] was the first one to use an EM-based method for aggregating multiple answers from clinicians when recording a patient's medical history. Even being experts in their field, these annotators could provide different answers due to the difficulty of the questions posed in that process. Dawid and Skene model each annotator with a conditional probability distribution over the classes given the real label, which forms a $r \times r$ matrix α^a. Each element of the matrix α^a_{jk} represents the probability that annotator a labels an instance of true class j with class label k. Thus, that matrix models the behaviour of the annotator, their reliability and their tendency to confuse one class with another. Two cases are considered, (i) the ground truth labels are known and then the α^a_{jk} parameters can be estimated through frequencies, and (ii) the ground truth labels are not known. The EM algorithm is applied for the latter case to learn the parameters of the annotator models and label the examples simultaneously. In the E-step, where the class labels are estimated, the parameters of the annotator models serve to weigh the votes of the labellers according to their reliability. This method performs label aggregation only, the first stage of a sequential approach.

The sequential learning approach based on EM has been extensively used in crowd learning, inspired by the work of Dawid and Skene. The popular GLAD algorithm [25] uses a model that accounts not only for the reliability of the annotators, but also for the difficulty of the instances, and uses an EM-based method to estimate the value of its parameters as well as finding estimates for the ground truth. While in GLAD each annotator and each instance are modelled using a single parameter, Welinder et al. [41] considers a more complex model with more parameters for each of them. Their model takes into account that annotators may have larger expertise in certain areas or types of instances. They rely on an iterative EM-like procedure to estimate both the model parameters and the labels simultaneously, in binary image classification. Yang et al. [44] use a simpler model for task difficulty and annotator behaviour combined with an EM strategy, computing the probability that an annotator may know the true label of an instance. In [27], there is only an annotator model, but labellers are not assumed to have the same reliability over all instances. Thus, their annotator models take into account which instance is each annotator labelling. Zhang et al. [42] use an EM method and a model similar to that of Dawid and Skene; the difference lies in the initialisation, in which they use a spectral method to obtain initial estimates, outperforming random initialisation.

EM-based methods can also implement a joint learning scheme, where both the parameters of the annotator model and those of the classification model are estimated simultaneously. Among them, the early key work of Raykar et al. [13] stands out. It is based on an annotator reliability model for binary classification but is easily adaptable to multi-class. The annotator model uses only two parameters per annotator: i) parameter α_1^a represents the probability that annotator a marks an instance of true class 1 correctly, that is, the sensitivity of the labeller; and ii) parameter α_0^a represents the probability that annotator a labels an example of true class 0 correctly, that is, the specificity of the annotator. When extended to the multi-class case, their annotator model becomes similar to that of Dawid-Skene [24]. The main difference with the previous approach is that Raykar et al. propose a joint learning method where a classification model is learned at the same time as the annotator model, as opposed to Dawid and Skene's work.

As for other approaches to joint learning, Sheng et al. [29] presented several simple methods for joint learning. They weigh each instance or duplicate them according to the received labels and then use a cost-sensitive classifier to learn from the weighted examples. Rodrigues et al. [30] propose a method for learning deep neural networks from crowds. They develop an EM algorithm that allows them to learn the parameters of the annotator model and those of the neural network jointly. They also propose the use of a so-called crowd layer in the network, so that it can be trained directly from the crowdsourced labels using backpropagation. Tanno et al. [31] learn the classifier and the parameters of the annotator model jointly, adding a regularisation term to the loss function and then finding the parameters that minimise it.

1.4 Motivation, objectives and overview

The principal aim of this dissertation is to overcome some of the weaknesses detected when studying the state of the art in learning from crowds. We build on top of key previous works and propose three enhancements to make the learning process more effective and efficient, by considering sources of information already available with no additional cost.

First, the label aggregation step for sequential approaches is analysed. In the works described in Section 1.3 that deal with label aggregation, different types of voting are employed. The votes of the annotators are weighted according to their reliability, measured through a model that represents their behaviour. Most of these approaches only use the annotations provided by the annotators to train the model and compute the weights. The descriptive feature of the instances, which establishes similarities between the instances that belong to the same class, is often disregarded. It is used to model the annotators only in a few cases, but up to our knowledge, it has never been used as part of the aggregation procedure. This leads to our first objective:

Objective 1:

To develop an aggregation method that takes into account the information of the explanatory features too. To study how to integrate the information of the explanatory feature into the label aggregation process and to design an efficient method that profits from that idea to enhance label aggregation.

For this purpose, we propose *domain-aware voting* in Chapter 2. This approach preserves the simplicity of schemes like MV and incorporates information from the descriptive variable of the instances by means of an extra vote called *domain vote*. The use of this extra vote can enhance label aggregation, especially when there are ties between different options or there are few or even no labels available for a part of the dataset. Also, there is no extra cost involved as the domain vote is obtained through information already available: other instances from the dataset and their crowdsourced labels. We will analyse the functioning of this method, and study its benefits and weaknesses through a set of experiments with both real and synthetic data.

Secondly, we propose another improvement that has to do with extracting as much information as possible from the annotators. All of the approaches to learning from crowds mentioned in Section 1.3 are based on full labelling, i.e., each labeller is required to select a single class label for each data example. Considering that annotators are non-experts, even committed and upstanding labellers are expected to fail when attempting to provide the correct class label for an instance. Mistakes may occur because they are completely convinced that an actually erroneous answer is the right one. However, it is arguably more reasonable to think that they might be doubtful when selecting the label and have more than one option in their minds. In this latter case, requiring annotators to provide a single label forces them to make a guess (they could just select at random one of the options they are considering). Note that, thus, the information about the other options that the annotator was considering is lost. We hypothesise that, by letting annotators express their doubts, all that information that is lost when using full labelling can be taken into account for the label aggregation or learning process, which could benefit from it. Thus, our second objective is:

Objective 2:

To offer a flexible labelling scheme for obtaining more information from the annotators, and to study whether it constitutes an improvement compared to full labelling.

To fulfil this objective, in Chapter 3 we introduce a novel labelling scheme, called *candidate labelling*. It consists of allowing labellers to provide a set of labels (called *candidate set*) instead of a single one so that they can express their doubts. We hypothesise that more information could be obtained without requiring more labellers, just making a different request to the same set of annotators. Adapted to that type of labelling, an annotator model and two different label aggregation methods are proposed. The proposed aggregation techniques are (i) a simple method that can be seen as the generalisation of MV to the candidate labelling scheme, and (ii) an EM-based algorithm that uses the annotator model we propose. The performance of these methods is studied through a variety of experimental scenarios using simulated annotations on real supervised datasets. The accuracy of the new voting with candidate labelling is fairly compared to that of MV with full labelling too, using a common synthetic label generation procedure. Employing candidate labelling with the simple voting method seems to enhance the results obtained using full labelling and MV, especially in scenarios where it is difficult to obtain a good performance. Also, the use of the more sophisticated EM-based method presents better performance than that of the voting approach.

Finally, in this thesis, we consider that an alternative labelling scheme, such as the candidate labelling, can ultimately positively impact the performance of classification models learned from it. We also aim to create learning methods that make use of candidate labelling, so our last objective is:

Objective 3:

To study how candidate labelling can be exploited for improving learning from crowds, and to develop techniques, both sequential and joint, that can learn from such data.

Chapter 4 presents the study of this question. We analyse the problem and propose two novel methods for learning with candidate set-based crowd-sourced labels. They are both based on the EM strategy. One of them, called *SL-C*, follows a sequential learning scheme, that is, estimates the ground-truth labels before learning the classifier. The other one, named *JL-C*, is a joint learning method that learns the classifier and aggregates the labels simultaneously. Their performance is tested through an extensive set of experiments and a wide variety of scenarios, using synthetic labels on both supervised datasets and partially labelled datasets. The learning techniques proposed for the candidate labelling framework seem to require fewer or less reliable labellers than their analogous methods from the full labelling scenario for obtaining similar results, showing an improvement in efficiency.

In the following chapters, we present the studies where these questions are consecutively addressed. In Chapter 5 we extract general conclusions from the different studies that form this thesis, and possible directions for future work are discussed. Finally, all the publications and other achievements of this thesis are catalogued.

2

Use of the Descriptive Variable for Enhancing the Aggregation of Crowdsourced Labels

2.1 Introduction

The use of crowdsourcing for annotating data has become a popular and cheap alternative to expert labelling. An aggregation process is usually required for combining the different labels provided by the annotators and providing a single one per example. This process, by the nature of crowdsourcing, is usually focused on being cost-effective, that is, to reach the maximum accuracy by counting on the minimum resources.

However, surprisingly, almost every label aggregation procedure from the literature determines the aggregated class value for an instance x using only its associated set of labels $\mathcal{L}_x$. The descriptive information provided by the explanatory variable of the instances, available in every machine learning problem, is rarely used to enhance label aggregation. Some approaches use it mainly for estimating the reliability of the annotators [13, 31, 45]. Sheng et al. [12] consider it to model the difficulty of the instances within a framework of active learning [35]. Zhang et al. [28] use the features and the labels to generate clusters in two layers, that are finally related to the true class labels. But in general, the contribution of the descriptive information into the final aggregation is indirect and hardly measurable. Similarly, given an observed instance, the rest of the dataset is usually not taken into account explicitly for label aggregation. Our intuition is that useful information for label aggregation can be inferred from other instances through the descriptive variable, assuming that the class conditional distribution evolves smoothly with respect to the instance space.

In this chapter, we propose *domain-aware voting (DAV)*, an extension of MV that carries out label aggregation by efficiently combining the labels available for the example at hand and using the information provided by its descriptive feature to gather information from the rest of the instances of the dataset. Thus, it can produce the correct labelling even when an example has never been annotated. A simple way to understand our proposal is to think of the k-Nearest Neighbours classifier [46]: it predicts a class distribution based on the neighbours of an example. In our framework, in terms of a nearest-neighbour approach, the annotations provided by the annotators for nearby examples would be aggregated to form a class distribution which will be added as an extra vote to the label aggregation process for a given instance. Nevertheless, DAV is a general solution that exploits the domain information by using an estimate of the class conditional distribution. This may be obtained in diverse ways. The domain information is used as an extra vote that is obtained at zero cost. DAV would be useful for sequential learning, as this technique can be combined with any classical learning algorithm after the aggregation step. Our proposal shares a similar level of simplicity to MV, as it does not build an annotator model. In our extensive empirical validation, DAV performs equal to or better than MV in most scenarios with a small number of labels for each instance. That is, DAV outperforms MV in terms of cost-effectiveness and its use can lead to reducing costs of labelling through crowdsourcing, which is the ultimate goal of resorting to a crowd of annotators for labelling.

The rest of the chapter is organised as follows. Firstly, our proposal is presented formally. Then, the behaviour of DAV, its ability to change the choices made by MV and its expected accuracy in diverse scenarios are studied. In Section 2.3, the hypotheses behind our method are tested through an extensive experimental setting, and the results are broadly discussed next. In Section 2.4 we elaborate on practical criteria for the use of DAV. Finally, we draw some conclusions and suggest open questions for future work.

2.2 Domain-aware voting

In this section, we present an extension of the classical MV approach which makes use of the explanatory variable X to incorporate information regarding all the instances from the dataset in the label aggregation process. The use of that information can help reduce the class uncertainty of each instance. A certain level of uncertainty is always intrinsic to any learning from crowds scenario, as the labels provided by a part of the annotators may be incorrect, and the number of gathered labels for a part of the instances could be insufficient for obtaining acceptable results with MV [16]. The class uncertainty surrounding an instance decreases as the size of its associated crowdsourced labels multiset $\mathcal{L}_x$ increases (more annotations) and the number of distinct labels decreases (annotations concentrated on particular classes, and in the best case, on a single class). In instances with high class uncertainty, the information obtained solely from $\mathcal{L}_x$ may be insufficient. The incorporation of the descriptive information of instances into the voting could enhance the performance of label aggregation. Moreover, we can also incorporate into the aggregation task the intuition that examples with similar descriptive features might also share the same class.

Our proposal, called *domain-aware voting (DAV)*, exploits the descriptive information of an instance x for inferring its class label, along with the information provided by the multiset of labels $\mathcal{L}_x$. DAV is expressed formally as follows,

$$\nu^*(x, \mathcal{L}_x) = \arg\max_{c \in \Omega} v^*(c|x, \mathcal{L}_x), \tag{2.1}$$

where the DAV estimate, $v^*(c|x, \mathcal{L}_x)$, is given by

$$v^*(c|x, \mathcal{L}_x) = \frac{1}{|\mathcal{L}_x| + \omega_0} \left[|\mathcal{L}_x| \cdot v(c|\mathcal{L}_x) + \omega_0 \cdot q(c|x) \right]. \tag{2.2}$$

The $q(c|x)$ is an estimate of the conditional class distribution $p(c|x)$, which we will call *domain vote.* It is an extra vote added to the voting estimate, which is weighted by means of a *weighing parameter* ω_0. DAV becomes the MV strategy when $\omega_0 = 0$. Interestingly, the DAV estimate has self-regulatory properties for the aggregation of annotations. Given a fixed ω_0 value, the influence of the domain vote in DAV decreases as the size of the set of labels $\mathcal{L}_x$ increases. In other words, as the number of collected labels tends to infinity, the DAV estimate tends to be the voting estimate. Similarly, as the size of $\mathcal{L}_x$ is reduced, the information provided by the domain vote gains relevance. This self-regulatory behaviour is particularly suitable for crowd labelling scenarios in which the size of the sets of labels of the different instances are typically unbalanced.

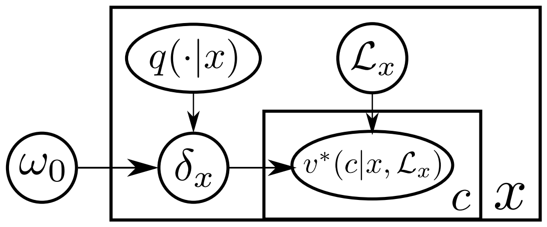


Fig. 2.1: Bayesian interpretation of DAV: For each instance x, ω_0 and $q(c|x)$ provide the hyper-parameters of the Dirichlet prior for that instance, δ_x. That Dirichlet prior, along with the labelling $\mathcal{L}_x$, provides the posterior probabilities for $v^*(c|x, \mathcal{L}_x)$, for each $c \in \{1, \ldots, r\}$.

In fact, assuming that $q(c|x)$ is given a-priori, DAV can be understood as a Bayesian estimate of the class distribution for an instance (see in Figure 2.1 its plate model), where domain votes are the hyper-parameters of a Dirichlet prior, $\delta_x = \omega_0 \cdot q(c|x)$. In this viewpoint, the weighing parameter ω_0 is the equivalent sample size, which determines the strength of our belief in the previous knowledge represented by q.

Note that DAV is a general method where the domain votes can be obtained through a variety of means: They could be considered as priors, the output of a classification model, or obtained through density estimation, to name a few. Throughout the remainder of the paper, we call *domain voter* to a classifier learnt with the voting estimates of the instances as probabilistic ground truth, which introduces the descriptive information into the label aggregation task. A classifier that can provide a probability distribution over the class labels is preferred, to reflect the uncertainty of its predictions. Depending on the specific application, our domain voter could be any type of classifier: from a simple naive Bayes for structured data to a deep neural network for image classification.

Note that, conceptually, DAV is aligned with the objectives of crowdsourcing, which was introduced as a cost-saving alternative to expert supervision. DAV considers an extra weighted vote which is obtained *for free*. While the aggregation performance is preserved, DAV requires fewer annotators, thus reducing the cost.

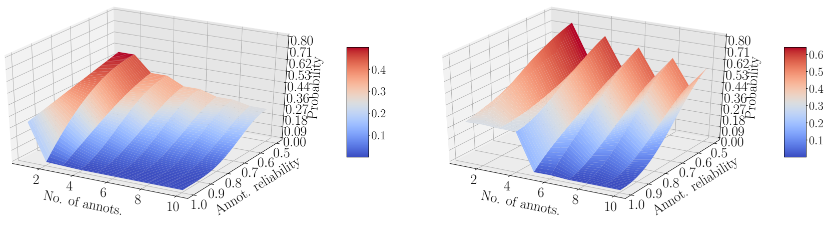
2.2.1 Intuition on the behaviour of DAV

For the sake of a better understanding of the expected performance of DAV, some insights into its behaviour under different conditions are given hereafter. Here we focus on two types of scenarios determined by annotator reliability and domain voter performance: (i) Scenarios in which the domain voter may switch the choices made by MV, and (ii) scenarios in which DAV is expected to obtain higher accuracy than MV.

For the sake of simplicity, let us consider a binary class ($r = 2$) and a deterministic domain voter ($q(c|x) = 1$ for a label c, and 0 for the rest of labels). Let us define annotator reliability or performance, α, as the probability that each annotator selects the correct label. In this binary class context, the most reliable annotator ($\alpha = 1$) always makes the correct choice, meanwhile, the least valuable contribution comes from those that randomly guess a label ($\alpha = 0.5$). As we have assumed that annotators provide on average the real class label, we have considered reliability values $\alpha \geq 0.5$.[1] The following results are based on the binomial distribution. Briefly, the probability that k annotators out of the total number m (all having reliability α) select the correct class label is $\binom{m}{k}\alpha^k(1-\alpha)^{m-k}$.

Firstly, it could be useful to have some insight into when the domain voter can shift the labels provided by MV. In Figure 2.2, the probability that the output of the domain voter differs from the one given by MV is depicted. The probabilities estimated for different numbers of annotators (from 1 to 14), reliability values (from 0.5 to 1) and ω_0 values (1 in Fig. 2.2a and 4 in Fig. 2.2b) are shown, using a domain voter with 0.7 of accuracy. According

[1] Remember that DAV does not model neither annotator reliability, nor any other characteristic. This reliability concept is an experimental design parameter.



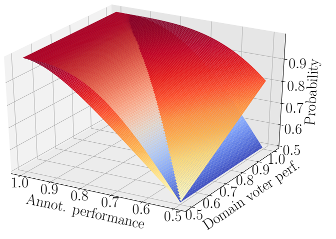
(a) $\omega_0 = 1$ (b) $\omega_0 = 4$

Fig. 2.2: Graphical description of the probability that the domain voter changes the choice made by MV, as the number of annotators increases (from 1 to 14) and the reliability of annotators increases (from 0.5 to 1). The value of the weighing parameter ω_0 is different for each subfigure and the performance of the domain voter is set to 0.7.

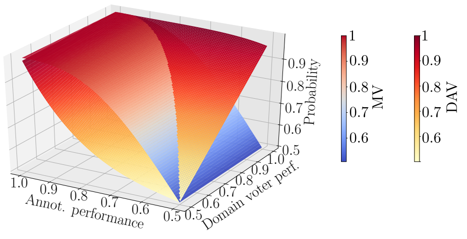
to Fig. 2.2, the probability that the domain voter changes the choices made by MV increases (i) as the number of annotators decreases and (ii) as their reliability decreases. On the one hand, as the reliability of the annotators decreases, a lower proportion of them will vote for the same label (a higher balance is expected). Thus, there is a higher probability that DAV tips the balance towards the other option. On the other hand, the expected difference between the number of votes gathered by both classes decreases as fewer annotators take part and, again, DAV has higher chances of giving an output different from that of MV. The aforementioned self-regulated behaviour of DAV can be observed: fixed the weighing parameter ω_0, the probability of shifting the decision of MV increases as the number of annotators decreases. Finally, note that, when the number of annotators is even, ties may occur when applying MV. In those cases, the domain vote would break the tie. This difference explains the stepped behaviour observed in Fig.2.2: the contribution of DAV is unquestionably more promising.

These results suggest that the reliability of the annotators has a greater influence than the number of annotators on the probability that DAV changes the answer of MV. The effective difference between both factors rises with large ω_0 values (Fig. 2.2b vs. Figure 2.2a). Reliable annotators (e.g. $\alpha = 0.9$) tend to concur voting for the correct label and, intuitively, shifting the choice made by MV is harder. Conversely, almost random annotators (e.g. $\alpha = 0.5$) tend to provide both labels at the same rate, and shifting the choice made by MV is more probable.

If the reliability of the annotators and the performance of the domain voter are known, the expected accuracy values of DAV and MV can be computed. That information would be useful to make decisions before applying DAV. In Figure 2.3 we compare both methods as the reliability of the annotators and the performance of the domain voter range from 0.5 to 1 and the weighing



(a) $\omega_0 = 1$.



(b) $\omega_0 = 4$.

Fig. 2.3: Graphical description of the expected accuracy values of DAV and MV, as the reliability of annotators increases (from 0.5 to 1) when the performance of the domain voter is equal to that of MV (ranging from 0.5 to 1). The value of the parameter ω_0 is different for each subfigure and the number of annotators is set to 5.

parameter ω_0 takes the values 1 and 4 (the number of annotators is set to 5). DAV is expected to outperform MV when annotators are unreliable and the performance of the domain voter is high. The difference increases as the reliability of the annotators decreases and the performance of the domain voter increases. MV outperforms DAV when annotators show intermediate reliability and the domain voter performs poorly. Note that usually a classifier trained with the results of MV as class labels is used as a domain voter, so it should perform better than the average annotator. Regarding ω_0, DAV outperforms MV more often in experiments where the weight of the domain voter is lower ($\omega_0 = 1$). However, the performance differences between DAV and MV are more prominent when the domain voter performs better and it is given a higher weight ($\omega_0 = 4$). Note that *not all* the scenarios observed in Fig. 2.3 are necessarily realistic. It is reasonable to expect that the domain voter performs better than a single annotator, as it might simply be built taking into account the labels provided by all annotators. A domain voter with a performance much lower than annotator reliability might be unusual in practice.

In the next section, we present a more realistic and extensive comparison between DAV and MV under varying experimental conditions.

2.3 Empirical study

The presented label aggregation scheme, DAV, is proposed as an enhancement of MV that incorporates extra information, from the descriptive variable and all the instances. We simulate a large spectrum of scenarios and aim to identify those in which DAV outperforms MV. Scenarios where instances might be labelled by few or no annotators, where these have varying reliability values,

Name	n	d	r	Name	n	d	r
Arrhythmia	452	279	13	*Segment*	2310	19	7
Dermatology	366	34	6	*Vehicle*	846	18	4
Glass	214	9	6	*Vowel*	990	10	11
Satimage	6435	36	6				

Table 2.1: Selected supervised datasets from UCI repository [47]. The columns display, in the following order: Name of the dataset, number of instances (n), dimension of the explanatory variable (d) and number of classes (r).

are considered. Each experiment is run 100 times, and the mean values of the accuracy are reported.

Our hypotheses are:

- **H1**: There exists an $\omega_0 > 0$ for every dataset that makes DAV better (or at least not worse) than MV in terms of aggregation accuracy (Eq. 1.1),
- **H2**: the advantage of DAV regarding MV tends to increase as the number of labels collected for each instance decreases, and
- **H3**: the advantage of DAV regarding MV increases as the reliability of the annotators decreases.

We validate these hypotheses with (i) standard supervised data and synthetic annotators, and (ii) real crowdsourced data in the following subsections.

2.3.1 Experiments with artificial annotations on standard supervised datasets

Firstly, we consider fully supervised datasets and synthetically transform them into crowdsourced labelled datasets employing simulated annotations. This allows us to control the reliability of the annotators, and thus to validate Hypothesis H3.

We consider datasets with different numbers of instances, class labels, and dimensions of the explanatory variable, to cover a variety of experimental scenarios. In that way, the strengths and weaknesses of DAV concerning the baseline MV, can be observed accounting for a wide range of characteristics. The datasets, collected from the UCI repository [47], and their main characteristics are summarised in Table 2.1.

2.3.1.1 Artificial labels generation

To generate meaningful synthetic labels for each dataset, we take into account the class-confusion matrix of a Random Forest (RF) classifier [48]. A reliability parameter (α) sets the probability that an annotator labels an instance correctly, and is used to simulate the mistakes of the annotators. As α tends to 1, annotators tend to make fewer mistakes on average. Recall that the reliability is not estimated in the studied label aggregation processes (MV and DAV), however, we use it as a parameter to control the performance of the annotators in this experimental setting.

To generate the multiple noisy labels, the following procedure is carried out. Given a supervised dataset, we use stratified 10-fold cross-validation [49] to estimate the mean class confusion matrix M of a RF model learned from it. The rows of M are normalised so that they all add up to 1. Then, a matrix ρ is constructed as follows. For $c \in \{1, \ldots, r\}$:

- $\rho_{c,c} = \alpha$
- For $c' \neq c$:
  - $\rho_{c,c'} = \frac{M_{c,c'}(1-\alpha)}{\sum_{c'' \neq c} M_{c,c''}}$, if $\exists c'' \neq c$: $M_{c,c''} > 0$.
  - $\rho_{c,c'} = \frac{1-\alpha}{r-1}$, otherwise.

In this way, the element $\rho_{c,c'}$ is the probability that an annotator assigns the label c' to an instance of real class c. The annotator model is consistent with the specified annotator reliability, as $\alpha = \rho_{c,c}$, and with the confusion between classes estimated in matrix M.

An annotation for an instance of class c_x can be simulated by sampling the distribution $\rho_{c_x} = (\rho_{c_x,1}, \ldots, \rho_{c_x,r})$. To obtain several artificial annotations, the distribution ρ_{c_x} is independently sampled. As our goal is not to model the annotators, we do not consider differences between them: all of them are simulated through the same matrix, ρ. Also, for the sake of simplicity, the same number of labels, l, is sampled for each instance. Given an instance x with real class c_x, the distribution ρ_{c_x} is sampled l times, and the obtained labels form the labelling $\mathcal{L}_x$.

2.3.1.2 Label sets of different sizes

Crowdsourced datasets usually have instances with different numbers of labels (some even with very few labels or none), a scenario strongly related to hypothesis H2. To consider this in our experiments, the label sets of the instances might be transformed in three different ways:

Config. A The datasets are used with all the sampled labels.
Config. B All labels assigned to a specific subset of the instances are discarded.
Config. C Labels are randomly discarded (uniformly or not).

For configuration B, the proportion of instances whose assigned label sets are emptied is controlled by a parameter p_d. In practice, labels are discarded as follows: An instance is randomly selected with probability p_d. Next, all the labels of the selected instances are discarded. The expected number of instances whose labels are discarded is $|D| \cdot p_d$. By assigning different values to p_d, the robustness of the methods in front of datasets with unlabelled examples can be observed.

For configuration C, a concentration parameter (con) controls the variance of the number of discarded labels for different instances. The proportion of labels to eliminate for each particular instance is determined by a Beta distribution. In practice, labels are removed as follows: Given an instance x, each label in $\mathcal{L}_x$ is discarded with probability $B_x \sim B(con, con)$. Since the two parameters of the Beta distribution are equal, the expected average number of discarded labels is $\frac{|\mathcal{L}_x|}{2}$. When $con = 1$, all the numbers of labels to discard in the range $\{0, \ldots, |\mathcal{L}_x|\}$ have the same probability. As $con \to 0$, the number of eliminated labels tends to be extreme (closer to either 0 or $|\mathcal{L}_x|$), i.e., the variance tends to its maximum. As $con \to \infty$, the number of discarded labels gets closer to the mean $\frac{|\mathcal{L}_x|}{2}$, i.e., the variance tends to 0. By varying the value of the parameter con, scenarios where there is a fixed budget but the annotations are distributed throughout the instances in different ways can be observed.

2.3.1.3 Implementation of DAV

A. Domain voter building:

Three models have been selected as domain voters: k-Nearest Neighbours (k-NN), Logistic Regression (LR) and Random Forest (RF). The domain voter is trained using all the annotated instances. In particular, the instances have probabilistic labels corresponding to their voting estimate (see Equation 1.3).

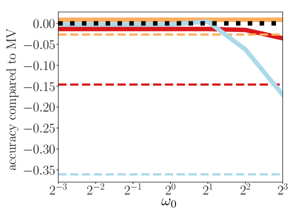
B. Operating DAV:

Given an instance x, the domain voter is used to get a distribution over the classes and the voting estimate (Eq. 1.3) is computed for all classes. Both are combined computing the DAV estimate as in Eq. 2.2, and the argument of the maximum is taken as the result (Eq. 2.1).

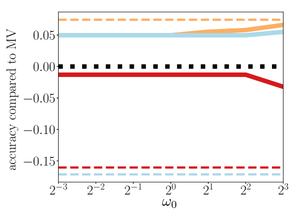
2.3.1.4 Experimental results with supervised datasets

The results obtained with supervised datasets (Table 2.1) and under different experimental conditions are discussed below. Inspired by real scenarios (see the following Section 2.3.2), we fix $l = 6$ simulated labels for each instance from the supervised datasets. Each experiment is run 100 times, and the mean values of the accuracies are obtained.

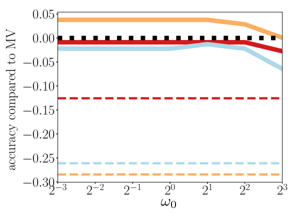
In Figure 2.4, the evolution of the mean accuracy with respect to the weight of the domain voter (ω_0) can be observed. The value of ω_0 ranges from 2^{-3} (when DAV closest resembles MV) to 2^3 in a logarithmic scale, without discarding any label (configuration A). The reliability parameter α is set to 0.7. DAV achieves a better (or at least equal) performance than MV in all



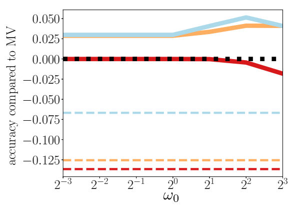
(a) *Arrhythmia*



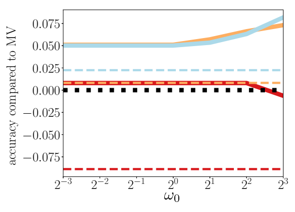
(b) *Dermatology*



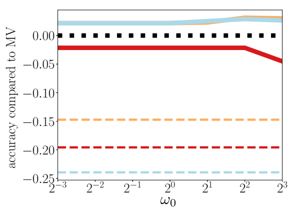
(c) *Glass*



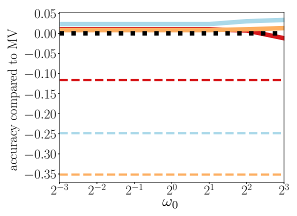
(d) *Satimage*

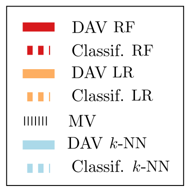


(e) *Segment*



(f) *Vehicle*

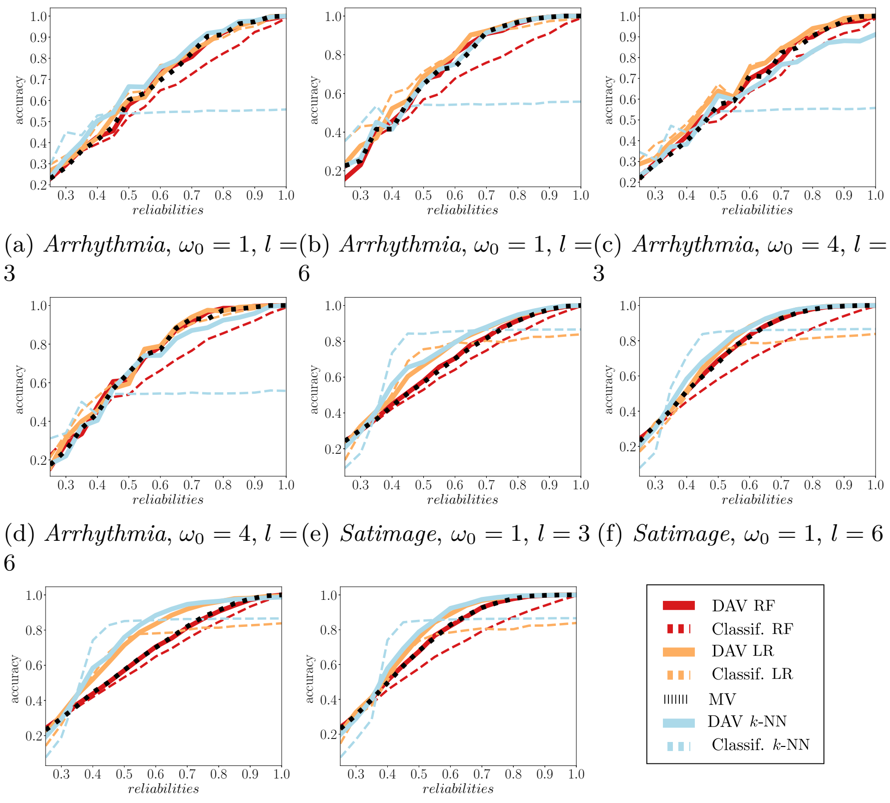




(g) *Vowel*

Fig. 2.4: Graphical description of the accuracy obtained by DAV with different classifiers and the classifiers themselves compared to the accuracy of MV, as the weight of the domain voter (ω_0) increases, $\omega_0 = 2^e$ with $e \in \{-3, \dots, 3\}$. Results obtained with artificial annotations on supervised datasets are displayed, using all labels (configuration A). The values of the rest of the parameters are fixed: $l = 6$ (number of labels per instance) and $\alpha = 0.7$ (reliability of the annotators).

the datasets, as there always exists a value of the weighing parameter ω_0 and a classifier for each dataset that allows DAV to outperform MV. Summing up through the different combinations of datasets and classifiers, DAV outperforms MV in 19 out of the 21 experiments. Note that DAV obtains higher accuracy than the domain voter in 20 out of the 21 experiments. When a classifier obtains a lower accuracy than MV, in most cases, the accuracy of DAV gets closer to that of MV as the weight of the domain voter increases. However, there are cases where the accuracy of DAV increases as the weight of the domain voter increases, such as the datasets *dermatology* (Fig. 2.4b), *satimage* (Fig. 2.4d) and *segment* (Fig. 2.4e). Thus, by using a selection criterion for the value of ω_0 (as discussed in Section 2.4.2), a setup that leads



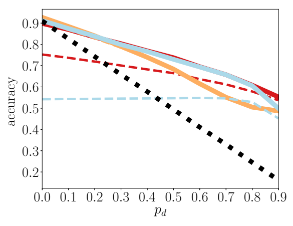
(a) *Arrhythmia*, $\omega_0 = 1$, $l = 3$ (b) *Arrhythmia*, $\omega_0 = 1$, $l = 6$ (c) *Arrhythmia*, $\omega_0 = 4$, $l = 3$

(d) *Arrhythmia*, $\omega_0 = 4$, $l = 6$ (e) *Satimage*, $\omega_0 = 1$, $l = 3$ (f) *Satimage*, $\omega_0 = 1$, $l = 6$

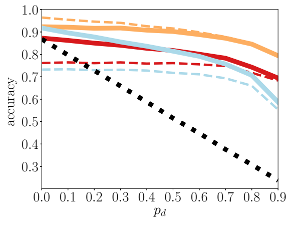
(g) *Satimage*, $\omega_0 = 4$, $l = 3$ (h) *Satimage*, $\omega_0 = 4$, $l = 6$

Fig. 2.5: Graphical description of the accuracy obtained by MV, DAV with different classifiers and the classifiers themselves, as the value of the parameter α (reliability of the annotators) increases, $\alpha \in \{0.25, 0.3, \ldots, 1\}$. Results obtained with artificial annotations on the supervised datasets *arrhythmia* and *satimage* are displayed, using the complete labellings (configuration A). Specific configurations (dataset and values of $alpha_0$ and l) are used in each subfigure, as detailed in their captions.

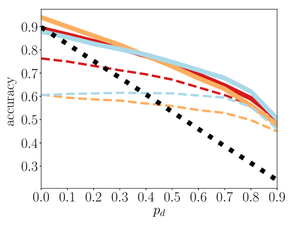
to equal or better performance than that of MV can be achieved. These results are in line with our Hypothesis H1. Note that the domain voters have a poorer performance than MV in almost all the scenarios observed in Figure 2.4. Nevertheless, DAV is still able to outperform MV in most cases: The extra information incorporated by DAV seems to complement the plain aggregation of labels. Moreover, DAV used with the k-NN model leads to the best results in almost all experiments, even though that classifier has an overall poorer performance than the other ones.



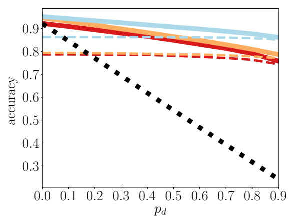
(a) *Arrhythmia*



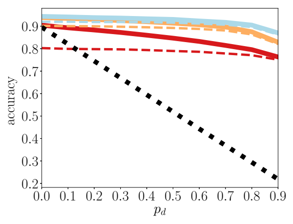
(b) *Dermatology*



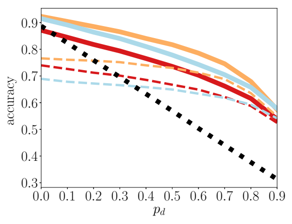
(c) *Glass*



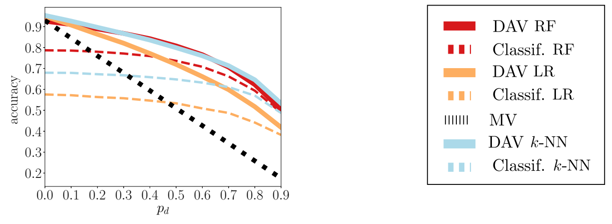
(d) *Satimage*



(e) *Segment*



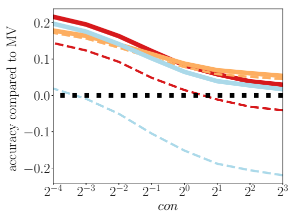
(f) *Vehicle*



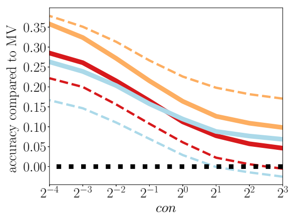
(g) *Vowel*

Fig. 2.6: Graphical description of the accuracy obtained by MV, DAV with different classifiers and the classifiers themselves, as the value of the parameter p_d (configuration B) increases, $p_d \in \{0, 0.1, \ldots, 0.9\}$. Results obtained with artificial annotations on supervised datasets are displayed. The values of the rest of the parameters are fixed: $\omega_0 = 1$, $l = 6$ (maximum number of labels per instance) and $\alpha = 0.7$ (reliability of the annotators).

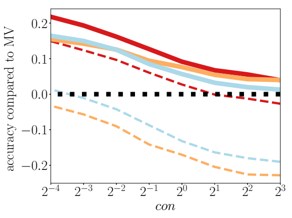
Figure 2.5 shows the evolution of the mean accuracy of the methods with respect to the reliability of the annotators, $\alpha \in \{0.25, 0.3, \ldots, 1\}$, and considering different values for parameters ω_0 and l. We concentrate in two datasets: *arrhythmia* and *satimage*, as they show similar trends to the results on other datasets). As the reliability of the annotators increases, so do the accuracy values of DAV and MV. The accuracies of DAV and MV are very similar for extreme values of α in most scenarios. DAV reaches a better performance than the domain voters for most levels of annotator reliability, except for the lowest values in *arrhythmia* dataset (Figs. 2.5a to 2.5d) and for medium values in *satimage* dataset (Figs. 2.5e to 2.5h). With *arrhythmia*, the reliability of the annotators does not have a visible influence in the differences between the



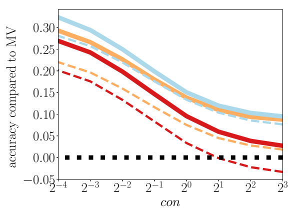
(a) *Arrhythmia*



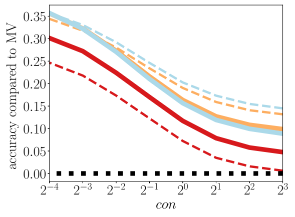
(b) *Dermatology*



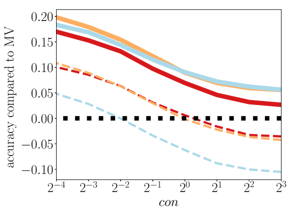
(c) *Glass*



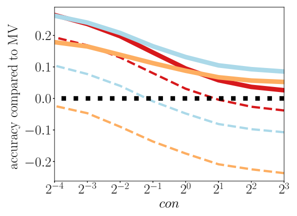
(d) *Satimage*



(e) *Segment*



(f) *Vehicle*



(g) *Vowel*

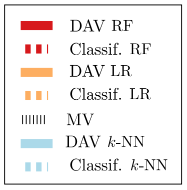


Fig. 2.7: Graphical description of the accuracy obtained by DAV with different classifiers and the classifiers themselves compared to the accuracy of MV, as the value of the parameter *con* (configuration C) increases, $con = 2^e$ with $e \in \{-4, \dots, 3\}$. Results obtained with artificial labels on supervised datasets are displayed. The values of the rest of the parameters are fixed: $\omega_0 = 1$, $l = 6$ (max. no. of labels per instance) and $\alpha = 0.7$ (reliability of the annotators).

accuracy values of the studied methods, as opposed to *satimage*. Moreover, in the cases where the reliability affects the difference between the accuracy values of DAV and MV, this increases quickly with low reliability values and then reduces smoothly. This behaviour is related to our hypothesis H3, as there is a greater difference for non-extreme low reliability (α) annotators. Similarly to the previous one, Figure 2.5 shows that the mean accuracy of the domain voter is lower than that of MV in almost all the experiments.

In Figure 2.6, configuration B (Sect. 2.3.1.2), where the label set of each instance is emptied with probability p_d, is studied. The values of p_d range from 0 to 0.9 and the value of ω_0 is set to 1, i.e., the domain voter has the same weight as any other annotator. As the proportion of non-annotated instances

(p_d) grows linearly, the performance difference between DAV and MV grows linearly as well, until the proportion of unlabelled instances reaches $0.5 - 0.7$. Then, in most cases, that difference slightly decreases, with a few exceptions (Figs. 2.6a and 2.6d). That is, DAV does not seem to be affected by the lack of labels as much as MV does, which supports our hypothesis H2. Note that, as the proportion of unlabelled instances (p_d) grows, the accuracy of each classifier gets closer to the accuracy of DAV obtained with that classifier. This behaviour is related to the fact that DAV provides the same label as the domain voter for unlabelled instances.

Results under experimental configuration C are displayed in Figure 2.7. The evolution of the accuracy with respect to the concentration of labels (con) (values 2^e where $e \in \{-4, \dots, 3\}$) can be observed. The rest of the parameters are fixed: $\omega_0 = 1$ and $\alpha = 0.7$. Let us recall the effect of parameter con in the distribution of labels: When the parameter con has low values, half of the instances tend to lose all their labels; and when con is high, all the instances tend to lose half of their labels. In this way, the effect of the lack of labels is observed in the whole spectrum between the two aforementioned scenarios. The average difference between the performances of DAV and MV observed in Figure 2.7 is greater than the one observed in Figure 2.4. This fact matches Hypothesis H2 since fewer labels are collected in average in configuration C (Fig. 2.7) than in configuration A (Fig. 2.4). Moreover, the difference between the accuracy values of the two methods is larger when a subset of instances is unlabelled (low values of con) than when all the instances provided have fewer labels (high values of con). Indeed, this is related to the self-regulatory behaviour of DAV: Given a weight for the domain voter (ω_0), the domain vote gains importance over the votes of the annotators as the number of available labels decreases. It is again noteworthy that, even when a classifier reaches a poorer performance than MV, DAV outperforms MV when using that classifier as a domain voter.

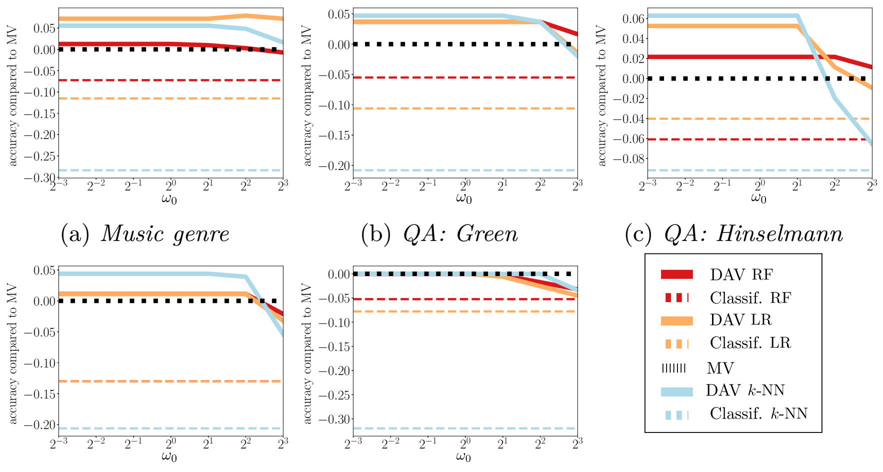
2.3.2 Experimental results with real-world crowdsourced datasets

In this second set of experiments, real crowdsourced datasets are used to test our hypotheses. Datasets with different numbers of annotators and mean numbers of labels per instance have been considered, as summarised in Table 2.2. A similar experimental setting as in the previous subsection is followed. It only differs in the fact that, in this new set of experiments, real crowd annotations are available and their simulation is not needed.

Figure 2.8 shows the evolution of the accuracy with respect to the weight of the domain voter (ω_0), which ranges from 2^{-3} (when DAV closest matches the behaviour of MV) to 2^3 in a logarithmic scale, without discarding any label (configuration A). According to Figure 2.8, H1 seems to be supported as DAV outperforms or at least equals the performance of MV for $\omega_0 \leq 2$ on all the considered datasets and classifiers. Moreover, the average difference between the accuracy values of the two methods seems to be higher. Again, the

Name	n	d	r	**# annot.**	**mean # labels**
music genre	700	124	10	44	4.21
QA: Green	98	62	2	6	6
QA: Hinselmann	97	62	2	6	6
QA: Schiller	92	62	2	6	6
Sentiment polarity	4999	1200	2	203	2.55

Table 2.2: Selected crowdsourced datasets. The columns display, in the following order: Name of the dataset, number of instances (n), dimension of the explanatory variable (d), number of classes (r), number of annotators and mean number of labels per instance. The dataset *music genre* is from [50], the datasets of *Quality assessment (QA)* are from the UCI repository and *sentiment polarity* was introduced by [51].

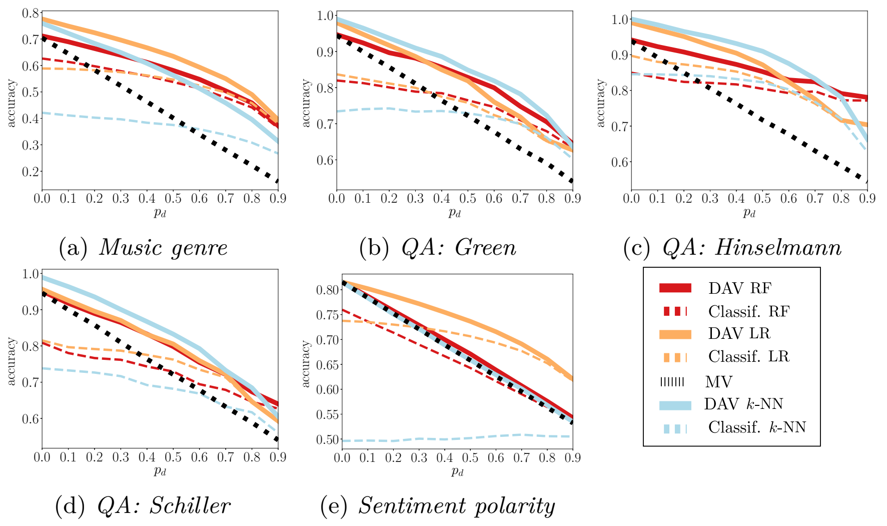


(a) *Music genre* (b) *QA: Green* (c) *QA: Hinselmann*

(d) *QA: Schiller* (e) *Sentiment polarity*

Fig. 2.8: Graphical description of the accuracy obtained by DAV with different classifiers and the classifiers themselves compared to the accuracy of MV, as the weight of the domain voter (ω_0) increases, $\omega_0 = 2^e$ with $e \in \{-3, \ldots, 3\}$. Results obtained with real crowdsourced datasets are displayed.

weight of the domain voter (ω_0) increases, it gains more importance over the crowdsourced labels, and the accuracy of DAV tends to that of the classifier. If the accuracy of the classifier is lower than that of MV, it may affect DAV resulting in a worse performance than MV. As aforementioned, the results suggest that an equal or better accuracy than that of MV can be achieved with DAV for certain values of the weighing parameter ω_0.



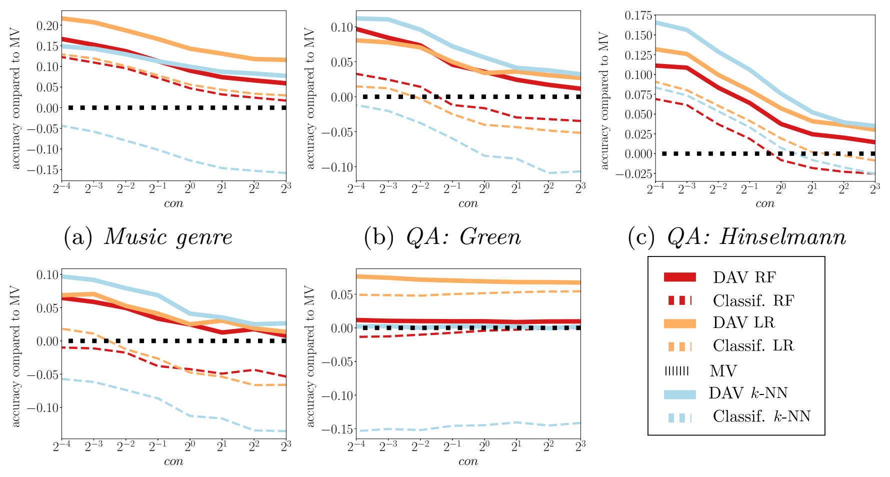
(a) *Music genre* (b) *QA: Green* (c) *QA: Hinselmann*

(d) *QA: Schiller* (e) *Sentiment polarity*

Fig. 2.9: Graphical description of the accuracy obtained by MV, DAV with different classifiers and the classifiers themselves, as the value of the parameter p_d (configuration B) increases, $p_d \in \{0, 0.1, \dots, 0.9\}$. Results obtained with annotations of real crowdsourced datasets are displayed. The value of the weighing parameter ω_0 is set to 1.

In Figure 2.9, the results for experimental configuration B (Section 2.3.1.2) are displayed, where all the annotations of each instance are discarded with probability $p_d \in \{0, 0.1, ..., 0.9\}$. Recall that, when $p_d = 0$, all the labels of each dataset are included. The rest of the parameters are fixed: $\omega_0 = 1$ and $\alpha = 0.7$. In that figure, similar patterns to those observed in the real crowd datasets can be seen (Fig. 2.6 in Section 2.3.1.4). The increase in the difference between the accuracy values of DAV and MV is almost linear with respect to the evolution of the parameter p_d, with a small drop for $p_d \geq 0.7$, in almost every scenario.

Configuration C is considered in Figure 2.10. Labels are discarded depending on a Beta distribution $B(con, con)$ as explained in Section 2.3.1.2 and the results are displayed for different values of the concentration of labels (con) (values 2^e where $e \in \{-4, \dots, 3\}$). The value of ω_0 is fixed to 1. The results match those observed in the experimental results obtained with artificial labels, although the differences between the accuracy values of DAV and MV are more limited in this case. A larger difference between the accuracy values of DAV and MV can be observed when there is a lack of labels than when all instances are provided $l = 6$ labels (Fig. 2.8), which would support our hypothesis H2. Furthermore, that difference increases when all labels are con-



(a) *Music genre* (b) *QA: Green* (c) *QA: Hinselmann*

(d) *QA: Schiller* (e) *Sentiment polarity*

Fig. 2.10: Graphical description of the accuracy obtained by DAV with different classifiers and the classifiers themselves compared to the accuracy of MV, as the value of the parameter *con* (configuration C) increases, $con = 2^e$ with $e \in \{-4, \dots, 3\}$. Values of *con* are 2^e, starting with $e = 3$ and decreasing to $e = -4$. Results obtained with annotations of real crowdsourced datasets (Table 2.2) and the weighing parameter $\omega_0 = 1$ are displayed.

centrated in a part of the dataset (low values of parameter *con*), which is a similar scenario to configuration B (Fig. 2.9).

Overall, the results obtained in this set of experiments are in line with those with synthetic data. Once again it is noteworthy that DAV outperforms MV even when its underlying classifier does not show better results than MV. Similarly, DAV obtains higher accuracy than the domain voter in all the studied scenarios.

2.4 Discussion

Our DAV method can be a promising tool for tackling label aggregation in learning from crowds environments. Evidence collected through two sets of experiments seems to support our three working hypotheses:

H1 Results in Figures 2.4 and 2.8 show that, for each dataset and classifier, there is at least one value of the weighing parameter $\omega_0 > 0$ such that DAV outperforms or equals the accuracy of MV.

H2 Results in Figures 2.6, 2.7, 2.9 and 2.10 show that the advantage of DAV over MV increases when there are fewer labels available.

H3 Results in Figure 2.5 show that there is a greater advantage of DAV over MV for (non-extreme) low-reliability values.

Two other noteworthy facts are that, in almost all of the observed cases, (i) DAV obtains an enhanced performance compared to that of the domain voter and (ii) DAV achieves better results than MV even when the domain voters do not.

When applying DAV, several decisions such as the method to obtain the domain votes or how to select the value for the weighing parameter ω_0 must be made. The ideal way of making those decisions would be by selecting the values that lead to the best performance of DAV. Unfortunately, this involves the estimation of the performance in the context of crowdsourced labelled data, which is an unsolved problem with short related literature (e.g., [52]). A few guidelines are offered below on the way of obtaining the domain votes and the selection of a value for ω_0, including other issues.

Some of these guidelines might require an *uncertainty* measure for quantifying how sure we are about the consensus label obtained for a given instance. One could use the entropy of the DAV estimate over the class labels of each instance, taking into account the number of collected labels. But this is not enough, as even though an instance with a single label would have entropy equal to 0, this label might be mistaken since annotators are not experts. One could, instead, perform Bayesian estimation using Dirichlet priors with all hyper-parameters equal to 1. Another option could be the Label and Model Uncertainty (LMU) proposed by [12]. In this framework, considering a binary class, the Label Uncertainty (LU) is computed as the tail probability below the labelling decision threshold, assuming that the posterior probability over the true label follows a Beta distribution whose parameters depend on the numbers of both positive and negative votes. The Model Uncertainty (MU) is a score that uses classifiers trained on the available data, and the LMU is computed as the geometric mean of the LU and the MU.

2.4.1 Construction of the domain votes

A key contribution to DAV comes from the domain voter. In the experiments presented in this work, the domain voter is a classifier. We suggest using the best available classifier in the state of the art for the domain of the problem at hand. Currently, all the instances are considered, with the same weight, to obtain the domain votes. However, one could use an uncertainty measure as aforementioned to identify certainly labelled examples. Instances with highly certain labelling could be given larger weight when building the domain voter, and the other way around. In the particular case that a subset of the instances is fully supervised (completely reliable), the domain voter could be obtained from this subset only. This is evident, for example, in the medical domain where intrusive practices such as punctures or biopsies are limited to a subset of patients. Techniques of semi-supervised learning [53, 54] could also be used to learn from a larger subset including the supervised examples. Finally, if the use of DAV reduces the uncertainty surrounding a specific subset of the instance space, the domain votes could be re-computed including that subset. This reveals a possible iterative application of DAV: The domain votes could be re-computed using the labels obtained through DAV, then perform DAV with the new domain votes, and so on.

2.4.2 Criteria for the selection of the weighing parameter

One of the main findings from our experiments is that the value of ω_0 is determinant and it has to be adjusted for the successful performance of DAV. There is no straightforward way to choose the optimal value for ω_0. As aforementioned, selecting ω_0 using cross-validation is unfeasible. Taking that into account, a few guidelines on the selection of the value of the weighing parameter ω_0 are as follows:

- Since ω_0 controls the weight of the domain votes on DAV, one could pay attention to the performance of the domain voter. When the performance of the domain votes increases, the value of ω_0 should be higher, and the other way around. If the performance of the domain votes can be estimated, it can help us make this decision.
- As the mean reliability of the annotators increases, the relative performance of the domain voter is reduced and a lower value for ω_0 could be chosen. In that case, the self-regulatory behaviour of DAV would cause a shift in the choices of MV only in instances with few labels or tied voting. Annotator models [24, 25, 26] could be used to estimate those reliability values.

As many of these concepts (good/bad performance, low/high uncertainty) are subjective, the final user has to choose among the considered scenarios and recommendations based on their judgement.

2.4.3 DAV in dynamic environments

Note that the scenario considered in this work is static: All of the instances and labels are available from the beginning. All of them are then used to obtain the domain voter, which is used to enhance the label aggregation process.

However, in many real-world applications, the environment is dynamic, i.e., new instances and/or labels may be gathered after the domain votes were computed. Different such examples include online learning, where instances come sequentially and not in a single batch from the beginning, and active learning [35], where new labels can be requested for specific instances. In these dynamic scenarios, the ideal strategy would be to re-compute the domain votes for every new piece of information (instance and/or label), as it is always beneficial for DAV. However, the methods for obtaining the domain votes could be excessively costly regarding the available resources. Thus, to adapt DAV to dynamic environments, one should consider whether the domain voter needs to be re-computed or not at every single step. To make that decision, one could use one of the aforementioned uncertainty measures to quantify the information gathered since the last update. For example, a new instance with low uncertainty or a new label that reduces the uncertainty of an instance would bring more information than an instance with higher uncertainty or a label which increases the uncertainty of an instance. When the amount of information brought by the new instances (or labels) is sufficiently high, the domain votes should be computed again including the new data in the dataset. The parameter ω_0 could be tuned accordingly as well.

2.5 Conclusions

In this chapter, a novel method for crowdsourced label aggregation called was presented, under the name domain-aware voting (DAV). As opposed to MV and other aggregation techniques, it uses information from the entire dataset and the descriptive variable for resolving the aggregation through an extra vote, weighted by the parameter ω_0 and distributed according to $q(c|x)$.

Empirical evidence, which was obtained through a vast experimental setting, supports our three hypotheses: (H1) there exists a weight for the domain vote for every dataset that makes DAV competitive regarding MV, (H2) the difference between the accuracy of DAV and MV is larger as the number of annotations per instance decreases, and (H3) the difference becomes bigger as the reliability of the annotators decreases. Thus, DAV arises as a useful alternative to MV, especially for scenarios where labels for each instance are scarce. DAV also exhibits an interesting self-regulated behaviour: The importance of the domain vote increases as the number of annotations decreases, and vice versa. As a consequence of the enhanced efficiency of DAV (its results are better with fewer annotations), the budget for crowdsourced labelling might be reduced. We also provide practical guidelines on how to set the weighing parameter of DAV. This work has been published in the *Knowledge and Information Systems* journal [55].

We have provided a method to improve efficiency in the procedure of aggregating labels provided by a crowd of annotators. The method is developed for the standard full labelling scenario, where each annotator provides one label for each instance. We believe that efficiency in learning from crowds can be further improved by proposing novel techniques not only for label aggregation or classifier learning but also for the labelling process. In the next chapter, we present the candidate labelling framework as an alternative to full labelling. Besides, we propose two aggregation methods derived from this new labelling scheme.

3

Label aggregation using candidate set-based labelling

3.1 Introduction

Crowdsourcing has become very popular among the machine learning community as a way to obtain labels. In most of the approaches that use crowdsourced labels (see Section 1.3), annotators are asked to provide, for each presented instance, a single class label (full labelling). Implicitly, the annotator is required to make a strong decision on each instance. However, the forced selection of a single class label can be too constraining. Considering that the labellers may not be experts, proceeding in this way could fail to take real advantage of the knowledge of the labellers. In that situation, the decision of an annotator might not reflect their whole knowledge about the given example (e.g., for difficult instances, scenarios with more than two classes or when the annotator is hesitant). For example, we will not know that the annotator was doubting between two (or more) class labels, even if these labels were equally plausible in their mind. Thus, it seems reasonable to consider a more flexible request than forcing them to provide a single label if we want to extract as much information as possible.

In the binary case of crowd learning, there have been previous attempts to soften the rigidity of full labelling. These usually provide an extra option so that the users can express their doubts and decide not to label an instance. For example, Torre et al. [56] introduce an “I do not know” option in citizen science, improving accuracy compared to the case without that option. In the Weather Sentiment - AMT dataset [1], annotators were asked to label the sentiment of tweets related to the weather, and an “I can’t tell” option was included. Ding et al. [57] consider that the self-confidence of an annotator and their ability are often related and allow them to use the “unsure” label, which seems to help reduce costs. In [58], the “unsure” option is provided within an active learning framework [35]. Differently, Smyth et al. [59] allow annotators to express their surety about the provided binary labels by selecting 1 out of

[1] www.kaggle.com/c/crowdflower-weather-twitter

5 surety levels. In [60], annotators are allowed to justify their response and express how sure they are about the provided label by submitting a short text along with the annotation.

In this chapter, the use of candidate labelling for learning from crowds is proposed: given an instance, the annotator is allowed to provide a set of candidate labels, called *candidate set*, instead of only one. This can be seen as a step forward in the direction of labelling flexibility, inspired by the partial labels (PL) problem [15]. PL is a weakly supervised problem [14] where each instance is associated with a set of candidate labels, with the guarantee that the real label is in that set. There is no restriction on the size of the sets of labels, which can vary from one instance to another. We extend this idea from supervised classification to the context of learning from crowds and allow annotators to provide as many labels as they want. In this context, however, the inclusion of the ground truth label in the candidate set is just a reasonable assumption, unlike in [15], where it is guaranteed.

The central idea of candidate labelling is that a more flexible request could allow for the extraction of more knowledge from the available annotators. For example, if an annotator is in doubt between two or more labels and they are forced to choose only one, they may pick the wrong one. On the contrary, if they are allowed to provide more than one label per instance, following the same example, the worker could select both. In this way, one might expect a lower number of mistakes, that is, they will include the correct class label in their sets of selected labels with high probability. Moreover, the fact that an annotator hesitates between a few class labels provides useful information about the underlying distribution of labels and the ground truth. Thus, more information could be extracted at the same cost. In other words, fewer crowdsourced labels could be required to obtain a similar level of information, saving costs in the labelling process. It may also involve less effort from the annotators: instead of spending more time deciding between two or more labels that they consider might be correct, they can keep all of them in the candidate set. The main intuition behind this study is that learning from crowds can be more cost-effective when annotators are allowed to provide this kind of labelling. In [61], they already provided some evidence that using candidate labelling (*checkbox interface* according to their terminology) can lead to faster and/or less costly labelling than using the traditional full labelling (*radio button interface*).

In social sciences, a labelling system similar to candidate labelling has been extensively studied under the name of *approval voting* (AV) [62, 63, 64]. Each user provides a set of labels for each instance as in the candidate labelling scenario. The aim is to aggregate the choices in surveys where there is no ground truth, identifying the popular (approved) options. All the approaches to approval voting consist of assigning to each instance the label that most annotators select. However, the goal of this research line is not focused on learning a classifier from the collected labels or any other model. In fact, in approval voting, the annotator who provides the largest set has the largest influence in the aggregation process. This could be detrimental when learning a classifier. In contrast, in the proposed candidate labelling, providing a larger candidate set indicates greater doubts about the annotation, which implies that the impact of the set in the estimated label is lower.

In the following, we explore the candidate labelling scheme for label aggregation, i.e., the first step of sequential learning approaches. The main hypothesis is that, by allowing candidate labelling, aggregation accuracy would be enhanced with respect to full labelling. In the remainder of this chapter, firstly the candidate labelling framework is formally proposed in Section 3.2 and the candidate voting technique is presented. Then, in Section 3.3, a data generation framework common to both full and candidate labelling is posed. Within that framework, the majority and candidate voting are compared throughout a variety of scenarios. Works that use the traditional full labelling model the reliability of the labellers based on the collections of labels they provide. In Section 3.4, we adapt the idea and present an annotator reliability model for the candidate labelling framework. Moreover, we propose an EM-based method that exploits this model for the aggregation of crowdsourced candidate sets of labels. Iteratively, the likelihood of the model parameters is maximised as the ground truth is estimated. The performance of that method is discussed in Section 3.6, based on experiments with synthetic labels created for real supervised data. Finally, conclusions about the results observed throughout this chapter are drawn.

3.2 Candidate labelling

Here we go beyond full labelling (see Section 1.2) to introduce the candidate labelling framework. The main novelty is that annotators $a \in A$ are allowed to provide a set of labels $L_x^a \subseteq \Omega_C$, called *candidate set*, for the instances $x \in \mathcal{D}$. In this chapter, we assume that $A_x = A$ for every instance and that $r = |\Omega_C| > 2$. These assumptions are not indispensable, the results gathered in this thesis can be extrapolated to scenarios where each instance is labelled by a subset of annotators. Also, in the binary case ($r = 2$), the use of candidate labelling would be equivalent to giving an “I do not know” option to the annotators [35, 56, 57, 58]. Candidate labelling can be considered a generalisation of that kind of option to the multi-class case, as it allows to express ideas like “I only know that it is *not* any of the labels I discarded” or “among the labels I provide, I cannot distinguish”. Indeed, annotators that include all class labels in the candidate set ($L_x^a = \Omega_C$) are expressing that they do not have any information. The candidate set L_x^a is assumed to include any class that annotator a considers plausible. Thus, depending on the difficulty of the instance x and the behaviour of the annotator a, the size of L_x^a could vary. Each instance x has associated multiple candidate sets $\mathcal{L}_x = \{L_x^a\}_{a \in A}$, provided by different annotators, and the labelling of the whole dataset is $\mathcal{L} = \{\mathcal{L}_x\}_{x \in \mathcal{D}}$.

In this chapter we focus on estimating the ground truth for a training set $\mathcal{D}$ given the candidate sets provided by the labellers, i.e., label aggregation. Thus, given $\mathcal{D}$ and $\mathcal{L}$, the goal is to obtain an aggregated labelling $\hat{\mathcal{L}}$ that maximises the accuracy described in Eq. 1.1 or, equivalently, minimises the (aggregation) *error*:

$$\epsilon(\hat{\mathcal{L}}) = \frac{1}{n} \sum_{x \in \mathcal{D}} \mathbb{1}(\hat{c}_x \neq c_x) = 1 - a(\hat{\mathcal{L}}). \tag{3.1}$$

For the candidate labelling setting, we first propose the use of *candidate voting* (CV) for estimating the ground truth. Given an instance x and the set of candidate sets $\mathcal{L}_x = \{L_x^a\}_{a \in A}$ gathered for it,

$$w_x(c) = \frac{1}{|A|} \sum_{a \in A} \frac{\mathbb{1}(c \in L_x^a)}{|L_x^a|} \tag{3.2}$$

is the *candidate voting estimate*. It generates a probabilistic labelling proportional to the weighted sum of annotators that assign label c to instance x, where each annotator a has a weight inversely proportional to the size of their candidate set, $|L_x^a|$.

We can apply a *winner-takes-all* strategy, taking the label that maximises the candidate voting estimate, to obtain a deterministic labelling:

$$\omega(\mathcal{L}_x) = \arg\max_c w_x(c), \tag{3.3}$$

The CV strategy can be understood as a generalisation of MV (Eq. 1.2) to the candidate labelling context. In fact, CV behaves as MV when all annotators provide a single label, $|L_x^a| = 1, \forall a \in A$. In practice, ties (i.e., when two or more class labels obtain the maximum number of votes) are solved randomly.

3.3 Comparison between full and candidate labelling

To study whether candidate labelling is more efficient than full labelling, we compare the standard voting from each labelling scheme, i.e., MV for full labelling and CV for candidate labelling. Efficiency is studied regarding the extraction of knowledge from a reduced number of annotators, especially when they doubt between several class labels. In this section, empirical evidence is provided in order to confirm the following closely related hypotheses:

- **H1**: candidate labelling requires (equal or) **fewer annotators** than the full labelling to achieve (equal or) lower error,
- **H2**: the number of annotators required by full labelling to achieve the performance of candidate labelling grows as **the difficulty of instances grows**,

- **H3**: candidate labelling stands out as the **number of possible class labels** increases,
- **H4**: candidate labelling stands out with **more hesitant annotators**.

The error of MV and CV will be estimated and analysed under different experimental conditions, which involve a varying number of annotators (together with their degree of hesitation), the difficulty of the instance, and the number of available labels. In this part, we have decided not to study the reliability of the workers. This characteristic would definitely have an impact on the results, obscuring the contribution of the candidate labelling. In order to explicitly control all the synthetic conditions and to develop simple and intuitive scenarios, experiments are carried out on artificial domains. Two types of simulations are performed: In one case (referred to as Case A hereinafter) the simulations are completely artificial, and, in the other one (Case R), they are derived from real-world datasets.

3.3.1 Data generation: Case A

In Case A, the difficulty of the instances and the ground truth are generated artificially. The experimental framework is graphically summarised in Fig. 3.1. It consists of four steps.
Step 0 - Parameters: The experimental framework takes two parameters for the generation of the artificial ground truth: δ, which controls the difficulty of the instances, and r, which is the number of possible class labels. There are three parameters that control the generation of the candidate sets: μ_0, which controls the degree of hesitation of the annotators, m, which is the number of available annotators, and s, which is the maximum size of the candidate sets, where $s = 1$ corresponds to full labelling and $s > 1$ to candidate labelling. The number of instances in $\mathcal{D}$ is fixed to $n = 500$ and the process of labelling is repeated 100 times for each instance. The selection of these values aims to ensure stable results that are barely affected by randomness.
Step 1 - Ground truth generation: Given a specific parametrisation, a domain is generated and represented by a Dirichlet distribution with hyperparameters $\delta_c = \delta$, for $c = 1, ..., r$. The Dirichlet distribution is used for sampling n probability distributions over the class labels, which represent the difficulty of the n instances. For each instance x, the difficulty distribution is denoted by $d_x(c)$, and the ground truth label corresponds to $c_x = \text{argmax}_c\, d_x(c)$. As $d_x(c)$ becomes more uniform, the difficulty of the instance increases. On the contrary, as $d_x(c_x) \to 1$, the difficulty of the instance decreases. Due to the properties of the Dirichlet distribution, on average, a high value of δ produces uniform distributions and, in consequence, difficult instances. On the contrary, a low δ value results in easy instances on average. After repeating this step for all the n instances, the set of ground truth labels $\{c_x\}_{x \in \mathcal{D}}$ is obtained. This is used in Step 4 for estimating the error.
Step 2 - Annotator simulation: The model of annotator used in this work consists of two parts: i) the behaviour of the annotator, which brings together

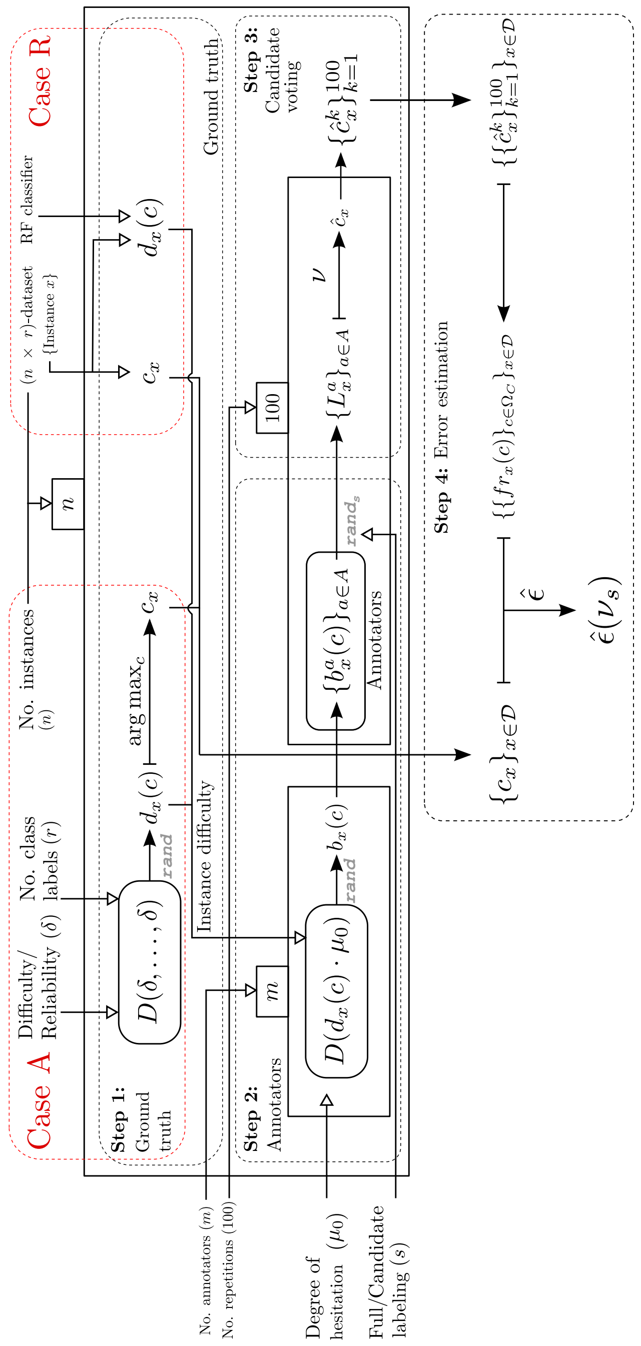


Fig. 3.1: Overview of the experimental framework with artificial domains.

their knowledge and the way they handle it, and ii) the labelling process given their behaviour. The behaviour is represented with a probability distribution over the class labels and the labelling process is simulated by means of random sampling.

For each instance, taking into account its difficulty, $d_x(c)$, a set A of m annotators is generated. As noted before, the behaviour of an annotator $a \in A$ regarding an instance is modelled by a distribution $b_x^a(c)$, where $b_x^a(c)$ represents the preference of annotator a towards the class label c. The distribution $b_x^a(c)$ is obtained by sampling a Dirichlet distribution with parameters $\mu_0 \cdot d_x(c)$, for $c \in \Omega_C$. Thus, the behaviour of an annotator depends on the difficulty of the instance $d_x(c)$ and the parameter μ_0. The parameter μ_0 can be seen as the average degree of hesitation of the annotators. For instance, as μ_0 tends to 0, the behaviour distribution will concentrate on a single label, that is, $b_x^a(c) \to 1$ for some class label c. Note that this does not mean that the annotator is right but only that he/she has a very low degree of hesitation. On the other hand, as $\mu_0 \to \infty$, the behaviour $b_x^a(c)$ becomes more similar to the difficulty $d_x(c)$. Reasonably, in no scenario does the behaviour of an annotator improve the instance difficulty.

Once the behaviour distributions of the annotators $\{b_x^a(c)\}_{a\in A}$ are fixed, the full and candidate labelling of the instance x are simulated. For this purpose, random sampling (with replacement) of size s of the distributions $b_x^a(c)$ is performed for each annotator $a \in A$. The parameter s controls the flexibility with which the annotators handle their knowledge to produce a candidate set —e.g., $s = 1$ corresponds to full labelling (no flexibility) while higher values of s correspond to candidate labelling (greater flexibility). Note that when the value of s increases, the probability that the correct class appears in a candidate set becomes higher, but the probability of selecting other classes also grows. All the distinct class labels sampled from $b_x^a(c)$ form the candidate set L_x^a of annotator a for instance x. Thus, the size of the candidate sets is upper-bounded by s.

Step 3 - Voting: Given the set of candidate sets $\mathcal{L}_x$ provided for instance x, an aggregated label is produced by means of the full (Eq. 1.2) or candidate voting (Eq. 3.3): $\hat{c}_x = \nu(\mathcal{L}_x)$. By repeating the labelling and voting processes 100 times, 100 estimated ground truth values are obtained, $\{\hat{c}_x^k\}_{k=1}^{100}$.

By repeating Steps 1 to 3 n times, the set of multisets $\big\{\{\hat{c}_x^k\}_{k=1}^{100}\big\}_{x\in\mathcal{D}}$ is obtained.

Step 4 - Error estimation: The goal of the experiments is to estimate and analyse in different settings the error of each voting technique for different values. Let us denote as $fr_x(c) = \frac{1}{100}\sum_{k=1}^{100} \mathbb{1}(\hat{c}_x^k = c)$ the *frequency* of the label c among the estimated ground truth values of instance x. The error of voting ν is estimated as follows:

$$\hat{\epsilon}(\nu) = \frac{1}{n}\sum_{x\in\mathcal{D}} 1 - (fr_x(c_x))$$

Additionally, following the methodology from [65], the estimated error $\hat{\epsilon}(\nu)$ is decomposed into the bias and variance terms:

$$\hat{\epsilon}(\nu) = \underbrace{\frac{1}{2n} \cdot \sum_{x \in \mathcal{D}} \left((1 - fr_x(c_x))^2 + \sum_{c \neq c_x}^{r} (fr_x(c))^2 \right)}_{\text{Squared bias}} + \underbrace{\frac{1}{2n} \cdot \sum_{x \in \mathcal{D}} (1 - \sum_{c \in \Omega_C} (fr_x(c))^2)}_{\text{Variance}} \quad (3.4)$$

3.3.2 Data generation: Case R

In real-world datasets, the difficulty distributions and the ground truth are available. Case A and Case R only differ in the number of parameters used (Step 0) and in the way the ground truth and the difficulty distributions are generated (Step 1). Three parameters are used in Case R: μ_0 (hesitation level of annotators), l (number of annotators) and s (flexibility). The number of instances n is fixed in each dataset. As in Case A, in order to ensure stable results, the labelling process is repeated 100 times for each instance.

Step 1 for Case R is as follows: Real data is used in order to get more realistic labels. The difficulties are obtained through random forest (RF) classifiers [48]. A fair validation is used to obtain class probabilities $p(c|x)$ for each instance x from an RF not trained with x. In order to avoid zero probabilities, we add a smoothing vector with values $1/r$ to the class probabilities. The resulting sum is normalised to obtain the difficulty distribution: $d_x(c) = (r \cdot p(c|x) + 1)/(2r)$. Note that, in this case, as opposed to Case A, when an instance is mistakenly classified by RF, we have $c_x \neq \arg\max_c d_x(c)$. The error achieved in a dataset by RF sets a lower bound for the error values that a voting method can reach.

Checking the hypotheses:

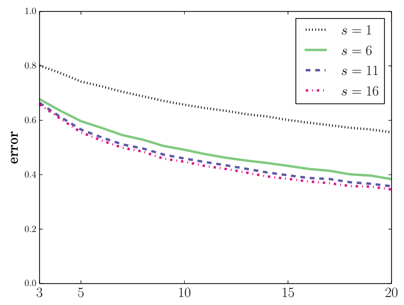
In this experimental framework, the soundness of our hypotheses can be checked by analysing the effects of different parameters in the estimated errors of full and candidate voting:

- **H1**: This hypothesis can be checked by using different values of the parameter m (number of annotators).
- **H2**: By varying the value of parameter δ (hyperparameter of the Dirichlet distribution), easier or more difficult instances can be generated and its effect on the performance of the different voting techniques can be observed.
- **H3**: Through the value of the parameter r (number of classes), the influence of the number of possible class labels can be tested.

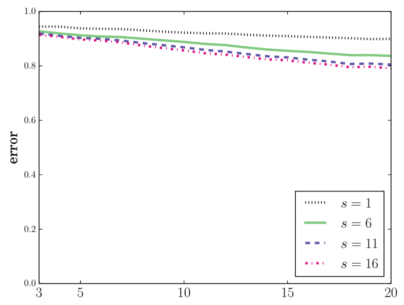
- **H4**: The parameter μ_0 allows us to control the degree of hesitation of the annotators to check this hypothesis.

3.3.3 Case A: Empirical results with artificial data

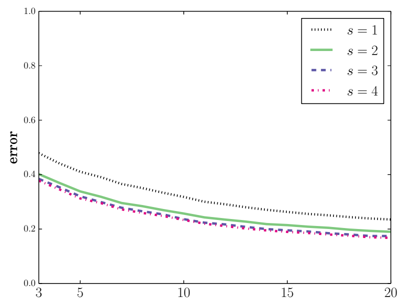
Using the framework proposed for Case A, a wide range of domains and annotators are generated by setting different values for the parameters r, δ, μ_0, s and m.



(a) $r = 32,\ \delta = 0.5,\ \mu_0 = 4$



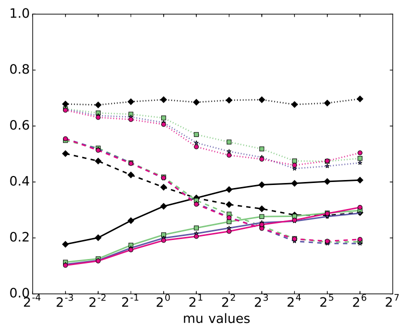
(b) $r = 32,\ \delta = 10,\ \mu_0 = 4$



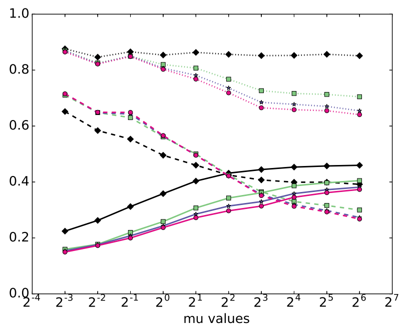
(c) $r = 8,\ \delta = 0.5,\ \mu_0 = 4$

Fig. 3.2: Graphical description of the error (Eq. 3.4) obtained by annotation simulated with different values of r and δ. Error curves for different values of s are shown in each figure.

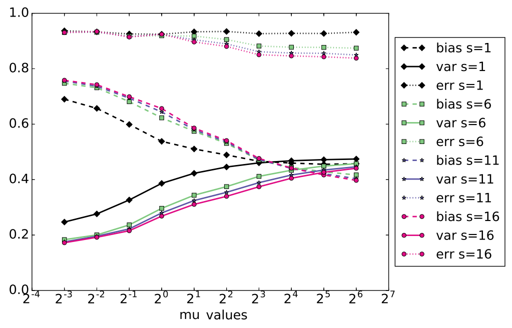
Experimental results are graphically summarised in Figs. 3.2 (total error) and 3.3 (bias/variance trade-off). Fig. 3.2 shows the results for the combination of difficult ($\delta = 10$) and easy ($\delta = 0.5$) instances, with different numbers of class labels $r \in \{8, 32\}$. In each plot, the parameters δ and r have been fixed at extreme values in order to observe the differences that they cause in the results. Each plot shows error curves obtained with different values of



(a) $r = 32,\ \delta = 0.5,\ l = 8$



(b) $r = 32,\ \delta = 2,\ l = 8$



(c) $r = 32,\ \delta = 10,\ l = 8$

Fig. 3.3: Graphical description of the decomposition of the error obtained by annotation simulated with different values of δ and s. Curves representing the squared bias, the variance and the total error are displayed in each figure.

$s < \frac{r}{2}$ and each curve shows the evolution of error as the number of annotators increases, $3 < m < 20$. For all three plots in Fig. 3.2, an intermediate value for the parameter that controls the hesitation of the annotators, $\mu_0 = 4$, has been selected in order to avoid its influence on the displayed results. In Fig. 3.3, the reader can observe the effect of increasing the hesitation of the labellers (parameter μ_0) in the bias/variance trade-off and analyse the source of the error in each scenario. In that figure, results with different difficulty degrees ($\delta \in \{0.5, 2, 10\}$), $r = 32$ possible class labels and $m = 8$ annotators are shown. Additional experiments with different settings have been carried out and the corresponding figures are available in Appendix A.

The error becomes lower as the number of annotators (m) increases in both the full labelling and the candidate labelling scenarios (see Fig. 3.2), i.e., more labellers provide more knowledge. Also, with both labelling approaches, better performance is observed in domains with a lower number of possible class labels ($r = 8$ in Fig. 3.2a as opposed to $r = 32$ in Fig. 3.2c) and in easier domains (see $\delta = 0.5$ in Figs. 3.2a and 3.3a as opposed to $\delta = 10$ in Figs. 3.2b and 3.3c).

According to these results, the full voting ($s = 1$) has consistently a poorer performance than the candidate voting ($s > 1$), which would support our hypothesis **H1**. For example, in the scenario displayed in Fig. 3.2c, full labelling needs at least 10 annotators to achieve the same error as the least flexible candidate labelling approach ($s = 6$ in Figs. 3.2a and 3.2b, $s = 2$ in Fig. 3.2c), and the error obtained by candidate labelling with $m = 5$ annotators is only achieved by full labelling with at least $m = 10$ annotators. Moreover, in Fig. 3.2a, the error shown by candidate labelling ($s \in \{6, 11, 16\}$) with $m = 5$ annotators is only observed with $m = 15$ annotators when using full labelling. In Fig. 3.2b, $m = 20$ annotators with full labelling perform as well as candidate labelling ($s \in \{6, 11, 16\}$) with $m = 5$ labellers.

The more difficult the domain is (i.e., the higher the parameter δ is), the larger the number of annotators required by full labelling in order to achieve a similar or lower error than candidate labelling (Fig. 3.2b vs. Fig. 3.2a). Moreover, in a difficult domain, full labelling barely profits from the increasing number of annotators (see Fig. 3.2b). For example, to achieve a similar error as 4 annotators with candidate labelling, in a difficult domain (Fig. 3.2b), 20 annotators with full labelling are required, while less than 15 annotators are sufficient in an easy domain (Fig. 3.2a). These last facts provide evidence for **H2**.

With a high number of possible class labels ($r = 32$), the difference between the error curves of full and candidate labelling (in all its different levels) becomes bigger, as hypothesised in **H3** (Fig. 3.2a vs. Fig. 3.2c). Many other experimental scenarios that show the same tendency and support both **H2** and **H3** are compiled in Appendix A.

As stated above, the effect of the parameter μ_0 on the error and the bias/variance trade-off can be seen in Fig. 3.3. When the hesitation (μ_0) of the labellers increases, the error curve of full labelling remains quite similar, while the error curves of different levels of candidate labelling decrease noticeably. In other words, similar results are obtained in full labelling with hesitant and obstinate labellers, while candidate labelling takes advantage of the hesitant labellers. Thus, in this experimental setting, the hypothesis **H4** holds.

A rise in the error curve of candidate labelling can be observed when there is an extremely high hesitation degree (large μ_0) along with easy instances (small δ). This can be clearly observed in Fig. 3.3a. In easy domains, the difficulty distribution usually assigns a much larger probability to a few class labels than to the rest. This can be interpreted as a dependence relationship between those highly probable class labels. Moreover, due to the high value of μ_0, that dependence also appears in the behaviour distributions. In that scenario, for sufficiently high values of s, all the candidate sets contain these highly probable class labels. Consequently, all these labels get the same number of votes, and the candidate voting results in a draw. As draws are solved randomly, candidate voting may be wrong even if the correct class label was selected in all the candidate sets.

As for the bias-variance trade-off, on the one hand, low values of μ_0 cause unbalanced behaviour distributions. On the other hand, high values of μ_0 lead to behaviour distributions that are similar to the previously generated difficulty distribution, which can be either unbalanced or uniform (depending on the value of δ). In the scenarios where the behaviour distribution is unbalanced, similar results tend to occur when performing the 100 repetitions of the sampling process. Thus, the mistaken guesses are concentrated in a few class labels, so the error is mostly caused by bias. That can be seen the plots of Fig. 3.3. The effect of high hesitation degrees (large μ_0) combined with easy instances (small δ) can be observed particularly in Fig. 3.3a. In scenarios with uniform behaviour distributions, different results are reached when repeating the sampling process, so variance becomes the main source of the error.

3.3.4 Case R: Empirical results with real supervised data

Dataset	**# instances**	**# attributes**	**# classes**	**RF error**
arrhythmia	452	279	13	0.334
vowel	990	10	11	0.353
segment	2310	19	7	0.026
letter	20000	16	26	0.059
mnist	60000	780	10	0.056

Table 3.1: Features of the datasets used for experiments: Number of instances, number of attributes, number of classes and error achieved with the Random Forest classifier.

In this set of experiments, five real-world datasets (displayed in Table 3.1) from the UCI repository [47] are used within the framework described in Section 3.3.1 for Case R. For each dataset, two different scenarios ($l \in \{4, 8\}$ annotators) are considered.

Experimental results can be observed in Table 3.2. For every dataset, the scenarios range from obstinate ($\mu_0 = 1$) to hesitant ($\mu_0 = 16$) annotators and varying flexibilities (s). The error reached with different values of the parameters s (flexibility) and μ_0 (level of hesitation) are compared in each scenario (fixed dataset and number of annotators), and the lowest error obtained in each scenario is highlighted in bold. For every dataset, the error obtained by RF in standard supervised classification is shown as a lower bound for the error achieved through both full and candidate labelling. As could be expected, the datasets in which RF obtains the lowest errors also show the lowest error values through majority voting and candidate voting.

As in Case A, the error also decreases when there are more annotators available (see $m = 4$ against $m = 8$). In all the cases with $\mu_0 > 1$ —except

Datasets	s	$m = 4$ $\mu_0 = 1$	$m = 4$ $\mu_0 = 4$	$m = 4$ $\mu_0 = 16$	$m = 8$ $\mu_0 = 1$	$m = 8$ $\mu_0 = 4$	$m = 8$ $\mu_0 = 16$
arrhythmia RF err: 0.334	1	0.622	0.619	0.620	0.514	0.514	0.512
	4	0.551	0.496	0.466	0.456	0.419	0.403
	7	0.538	0.472	**0.454**	0.444	0.408	**0.395**
vowel RF err: 0.353	1	0.638	0.639	0.640	0.542	0.543	0.543
	3	0.585	0.543	0.517	0.492	0.457	0.440
	5	0.572	0.516	**0.493**	0.479	0.440	**0.421**
segment RF err: 0.026	1	0.266	0.267	0.264	0.125	0.124	0.124
	2	0.202	0.160	0.135	0.084	0.062	0.054
	3	0.183	0.128	**0.099**	0.073	0.051	**0.043**
letter RF err: 0.059	1	0.376	0.377	0.377	0.191	0.192	0.191
	7	0.235	0.141	0.119	0.117	0.091	**0.083**
	13	0.205	0.128	**0.109**	0.086	0.085	0.085
mnist RF err: 0.056	1	0.369	0.369	0.368	0.216	0.215	0.215
	3	0.280	0.218	0.181	0.152	0.119	0.105
	5	0.258	0.184	**0.156**	0.138	0.105	**0.093**

Table 3.2: Error rates for experiments using different datasets and different values of the parameters μ_0, m and s. The lowest error obtained in each scenario is in bold. *RF err* refers to the classification error shown by a Random Forest classifier trained in the given dataset.

for $\mu_0 = 4$ with dataset *segment*— the error of candidate labelling ($s > 1$) is equal to or lower than that of the full labelling ($s = 1$), which supports our hypothesis **H1**. Hypothesis **H2** cannot be checked within this framework as the difficulties are pre-determined by the dataset and the performance of RF. Similarly, hypothesis **H3** cannot be contrasted because the number of classes r is fixed in each dataset, and its effect could not be isolated from that of other characteristics of the dataset (e.g., instance difficulty).

Experimental scenarios with a value of s near to $\frac{r}{2}$ show the best results. As the value of s increases, the error becomes lower in every case both with $m = 4$ and $m = 8$. This fact suggests that, when a labeller is more flexible, more information can be extracted. Note that candidate labelling ($s > 1$) always obtains a lower error than full labelling ($s = 1$), reaching error values similar to those obtained by RF on each dataset, i.e., close to the lower bound.

Similarly to Case A, candidate labelling ($s > 1$) profits from hesitant labellers (larger μ_0): The larger the value of μ_0, the lower the error. On the contrary, the error reached with full labelling ($s = 1$) barely changes from obstinate ($\mu_0 = 1$) to hesitant ($\mu_0 = 16$) annotators. Thus, as the hesitation level increases, candidate labelling outperforms the full labelling approach, which would support our hypothesis **H4**.

As a summary of Sections 3.3.3 and 3.3.4, our experimental results pose empirical evidence that supports our four hypotheses. Candidate labelling seems to gather more information of supervision than full labelling in crowd-sourced annotations. Candidate labelling is especially useful with a low number of workers, with difficult instances, with hesitant workers and/or with a large number of possible labels. As the trustworthiness of the labellers is not homogeneous, having information about their reliability can be of great advantage to aggregate the labels that they provide. In the next section, we extend the study within the candidate labelling framework by proposing an annotator model and a more sophisticated aggregation technique based on the EM algorithm.

3.4 Modelling annotators and maximum likelihood estimation

Obtaining an estimate of the ground truth or learning a classifier from the (probabilistic) labelling given by Equation 3.2 or 3.3 is possible. However, these implicitly assume that labellers show homogeneous reliability, which is usually not the case. When aggregating the labels gathered through candidate labelling, the contribution of each annotator can be weighted according to their reliability, and the accuracy of the aggregation process can be improved in that way. In this section, a model for the behaviour of annotators is described, with parameters that control their reliability and the way the candidate sets are generated. Then, the maximum likelihood estimates of the model parameters are inferred from the data.

In the presented model, the candidate set L_x^a is assumed to be generated by asking annotator a one question of the kind "Do you consider that the given instance x might belong to class c?" for each $c \in \Omega_C$. Let α_c^a denote the probability that annotator a includes label c in the candidate set for instances which really belong to class c. Let us also define β_c^a as the probability that annotator a includes any label $c' \neq c$ ($c' \in \Omega_C$) in the candidate set when annotating instances which really belong to class c. This implies that, given an instance of a certain class, the rest of class labels have the same probability of being mistakenly selected by annotator a. The parameters α_c^a and β_c^a provide us insights into the behaviour of annotator a when labelling instances that really belong to class c.

Assuming the candidate set generation process described above, the likelihood of the model given a candidate set L_x^a is:

$$\begin{aligned}Pr(L_x^a|\boldsymbol{\alpha},\boldsymbol{\beta},c_x) =&(\alpha_{c_x}^a)^{\mathbb{1}(c_x\in L_x^a)}\cdot(1-\alpha_{c_x}^a)^{1-\mathbb{1}(c_x\in L_x^a)}\\ &\cdot(\beta_{c_x}^a)^{|L_x^a|-\mathbb{1}(c_x\in L_x^a)}\cdot(1-\beta_{c_x}^a)^{r-|L_x^a|+\mathbb{1}(c_x\in L_x^a)},\end{aligned} \tag{3.5}$$

where $\boldsymbol{\alpha} = \{\alpha_c^a\}_{a\in A,c\in\Omega_C}$ is the set that groups the probability, for each annotator, of selecting the (unknown) correct label c_x, and $\boldsymbol{\beta} = \{\beta_c^a\}_{a\in A,c\in\Omega_C}$ is the set that groups the probability, for each annotator, of selecting each incorrect label.

Assuming that annotators provide the candidate sets independently and that all instances are i.i.d. according to $p(X,C)$, the likelihood given a dataset $\mathcal{D}$ and $\mathcal{L}$ is:

$$Pr(\mathcal{L},\mathcal{D}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{x\in\mathcal{D}}\prod_{a\in A} Pr(L_x^a|\boldsymbol{\alpha},\boldsymbol{\beta},c_x). \tag{3.6}$$

We can plug Equation 3.5 in this expression and account for all possible class labels, not only for the real class c_x:

$$\begin{aligned}Pr(\mathcal{L},\mathcal{D}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{x\in\mathcal{D}}\prod_{a\in A}\prod_{c\in\Omega}&\big((\alpha_c^a)^{\mathbb{1}(c\in L_x^a)}\cdot(1-\alpha_c^a)^{1-\mathbb{1}(c\in L_x^a)}\\ &\cdot(\beta_c^a)^{|L_x^a|-\mathbb{1}(c\in L_x^a)}\cdot(1-\beta_c^a)^{r-|L_x^a|+\mathbb{1}(c\in L_x^a)}\big)^{\mathbb{1}(c_x=c)}.\end{aligned} \tag{3.7}$$

We want to obtain the MLE of the model parameters, that is, the values of α_c^a and β_c^a that maximise the likelihood function from Equation 3.7. As the values that maximise the log-likelihood are the same that maximise the likelihood, we are going to compute the MLE by maximising the log-likelihood, which is given by the following expression:

$$\begin{aligned}\log\left[Pr(\mathcal{L},\mathcal{D}|\boldsymbol{\alpha},\boldsymbol{\beta})\right] = \sum_{x\in\mathcal{D}}\sum_{a\in A}\sum_{c\in\Omega}&\mathbb{1}(c_x=c)\cdot\Big(\mathbb{1}(c\in L_x^a)\cdot\log(\alpha_c^a)\\ &+(1-\mathbb{1}(c\in L_x^a))\cdot\log(1-\alpha_c^a)\\ &+(|L_x^a|-\mathbb{1}(c\in L_x^a))\cdot\log(\beta_c^a)\\ &+(r-|L_x^a|+\mathbb{1}(c\in L_x^a))\cdot\log(1-\beta_c^a)\Big).\end{aligned} \tag{3.8}$$

We then compute the derivative of the log-likelihood with respect to $\alpha_{c'}^{a'}$, for specific values $a'\in A$ and $c'\in\Omega_C$, and equal it to 0 to find its maximum:

$$\frac{d\log\left[Pr(\mathcal{L},\mathcal{D}|\boldsymbol{\alpha},\boldsymbol{\beta})\right]}{d\alpha_{c'}^{a'}} = \sum_{x\in\mathcal{D}}\mathbb{1}(c_x=c')\cdot\left(\frac{\mathbb{1}(c'\in L_x^{a'})}{\alpha_{c'}^{a'}}-\frac{1-\mathbb{1}(c'\in L_x^{a'})}{1-\alpha_{c'}^{a'}}\right)=0. \tag{3.9}$$

From this expression, the MLE of parameter α_c^a for any $a\in A$, $c\in\Omega_C$ is:

$$\alpha_c^a = \frac{\sum_{x\in\mathcal{D}}\mathbb{1}(c_x=c)\mathbb{1}(c\in L_x^a)}{\sum_{x\in\mathcal{D}}\mathbb{1}(c_x=c)}. \tag{3.10}$$

We proceed in an analogous way to find the MLE of parameters $\boldsymbol{\beta}$. We derive the log-likelihood expression of Equation 3.7 with respect to $\beta_{c'}^{a'}$, for specific values $a' \in A$ and $c' \in \Omega_C$, and equal it to 0:

$$\begin{aligned}\frac{d \log\left[Pr(\mathcal{L}, \mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta})\right]}{d\beta_{c'}^{a'}} &= \sum_{x\in\mathcal{D}} \mathbb{1}(c_x = c') \\ \cdot \left(\frac{|L_x^{a'}| - \mathbb{1}(c' \in L_x^{a'})}{\beta_{c'}^{a'}} - \frac{r - |L_x^{a'}| - \mathbb{1}(c' \in L_x^{a'})}{1 - \beta_{c'}^{a'}}\right) &= 0.\end{aligned} \tag{3.11}$$

From the previous equation, the MLE of parameter β_c^a for any $a \in A$, $c \in \Omega_C$ are:

$$\beta_c^a = \frac{\sum_{x\in\mathcal{D}} \mathbb{1}(c_x = c) \cdot (|L_x^a| - \mathbb{1}(c \in L_x^a))}{r \cdot \sum_{x\in\mathcal{D}} \mathbb{1}(c_x = c)}. \tag{3.12}$$

The estimate of the parameter α_c^a is the number of instances of class c for which annotator a included class label c in the candidate set over the total number of instances of class c. On the other hand, the estimate of the parameter β_c^a is the number of mistaken class labels that annotator a included in the candidate sets of all the instances of class c over the whole set of possible class labels for the total number of instances of class c.

The estimates in Equations 3.10 and 3.12 can be computed when the true class labels are known for all instances. Conversely, the true labels can be estimated given the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters using Bayes' Theorem:

$$Pr(c|\mathcal{L}_x, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto Pr(c) \cdot Pr(\mathcal{L}_x|\boldsymbol{\alpha}, \boldsymbol{\beta}, c). \tag{3.13}$$

Using Eq. 3.5 for the case that $c_x = c$ and estimating the marginal probability as $Pr(c) = \frac{\sum_{x\in\mathcal{D}} \mathbb{1}(c_x=c)}{|\mathcal{D}|}$, Equation 3.13 can be rewritten as:

$$\begin{aligned}Pr(c|\mathcal{L}_x, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto \frac{\sum_{x\in\mathcal{D}} \mathbb{1}(c_x=c)}{|\mathcal{D}|} \cdot \prod\nolimits_{a\in A} \Big(&(\alpha_c^a)^{\mathbb{1}(c\in L_x^a)} \cdot (1 - \alpha_c^a)^{1-\mathbb{1}(c\in L_x^a)} \\ &\cdot (\beta_c^a)^{|L_x^a| - \mathbb{1}(c\in L_x^a)} \cdot (1 - \beta_c^a)^{r - |L_x^a| + \mathbb{1}(c\in L_x^a)} \Big),\end{aligned} \tag{3.14}$$

where the marginal probability is given by $Pr(c) = \frac{\sum_{x\in\mathcal{D}} \mathbb{1}(c_x=c)}{|\mathcal{D}|}$. This probability distribution could be considered as an estimate for the ground truth.

In practice, neither the true labels nor the values of the parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are known. Thus, we have a mutual requirement for information that is indeed unavailable. This can be solved using a method based on the EM strategy [17] that estimates both the parameters of the model and the ground truth labels. In the following section, we propose an EM-based label aggregation technique for candidate sets.

3.5 EM-based method for candidate labelling aggregation

As mentioned in Section 1.2.3, the EM strategy attempts to gather maximum likelihood estimates when there is missing data. In the crowdsourcing context, the true class labels of the training instances are the missing data. EM is generally implemented as follows: First, an initial estimate of the ground truth labels is obtained. After that, the method consists of two steps: (i) **M-step**: The parameters that model the reliability of the annotators are updated with estimates that maximise or, at least, improve the likelihood achieved in the previous E-step; and (ii) **E-step**: Given an estimate of the parameters, the expected values of the ground truth labels are obtained for every instance, given the expected labels. The M and E steps are carried out iteratively until convergence. This method is guaranteed to converge to a local maximum except in rare cases.

Our proposal is an adaptation of the EM strategy to the scenario of learning from crowds with candidate labelling. Firstly, let us define $q(c|x)$ as the estimate of the probability $Pr(c|\mathcal{L}_x, \boldsymbol{\alpha}, \boldsymbol{\beta})$ described in Eq. 3.13, that is, the probability that x belongs to class c. In Equations 3.5, 3.10 and 3.12, the $q(c|x)$ estimates can substitute the expression $\mathbb{1}(c_x = c)$, switching from two discrete values (0 or 1) to any possible value in the continuous interval $[0, 1]$. Note that the true label c_x is unknown and this modification allows this approach to work with the probabilistic estimates of the ground truth. Our method works in the following way:

After a first step where the estimates $q(c|x)$ are initialised for all $x \in \mathcal{D}$ and $c \in \Omega_C$, the M and E steps of the proposed method are as follows:

- **M-step.** For every $a \in A$ and $c \in \Omega_C$, the parameters α_c^a and β_c^a are computed given q by means of Equations 3.10 and 3.12, using the estimates $q(c|x)$ instead of $\mathbb{1}(c_x = c)$.
- **E-step.** For every $x \in D$ and $c \in \Omega_C$, Equation 3.14 is used to compute the probability distributions $q(c|x)$ given the α_c^a and β_c^a parameters obtained in the M-step. As in the M-step, the terms $\mathbb{1}(c_x = c)$ are substituted by the previous estimates $q(c|x)$.

In the next section, the performance of the previously described method is tested. As there is no available crowdsourced data with candidate labels, we generate artificial labels. We profit from that synthetic labelling framework by controlling the different experimental parameters to observe the strengths and weaknesses of the proposed method in a variety of scenarios.

3.6 Experiments

In this section, the performance of the presented method is evaluated in different scenarios. In order to have insights into its performance: (i) the accuracy is computed for different scenarios, varying the numbers of annotators, classes, and instances, and the values of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters, (ii) the method is compared with candidate voting (Eq. 3.3), approval voting [62] and the **privileged aggregation** (where all $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters are known), and (iii) the evolution is observed through each iteration of the method.

3.6.1 Experimental setting

To the best of our knowledge, there is not any publicly available dataset for learning from crowds with candidate labelling. Thus, artificial data has been used as a means of obtaining experimental results. Simulated data is also useful to control the settings and explore different scenarios.

In order to generate different situations, the following experimental parameters are set to different values: number of instances (n), number of annotators (m), number of classes (r), minimum and maximum values of the $\boldsymbol{\alpha}$ parameters ($\underline{\alpha}$ and $\overline{\alpha}$) and minimum and maximum $\boldsymbol{\beta}$ parameters ($\underline{\beta}$ and $\overline{\beta}$). The parameters $\underline{\alpha}$ and $\overline{\beta}$ have both been fixed to 0.5, so that there always can be annotators of minimum expertise and adversarial annotators are not generated.

The method itself has two additional parameters:

- The convergence threshold δ. If $\frac{|\overline{\boldsymbol{\alpha}}_{(it)} - \overline{\boldsymbol{\alpha}}_{(it-1)}|}{\overline{\boldsymbol{\alpha}}_{(it-1)}} < \boldsymbol{\delta}$ or $\frac{|\overline{\boldsymbol{\beta}}_{(it)} - \overline{\boldsymbol{\beta}}_{(it-1)}|}{\overline{\boldsymbol{\beta}}_{(it-1)}} < \boldsymbol{\delta}$, where $\overline{\boldsymbol{\alpha}}_{(it)}$ ($\overline{\boldsymbol{\beta}}_{(it)}$) is the mean value of $\boldsymbol{\alpha}_{(it)}$ ($\boldsymbol{\beta}_{(it)}$) at iteration it, it is considered that the EM has converged. It has been set to $\delta = 0.05$.
- The smoothing parameter γ. There are two factors that lead to undesirable results, such as the likelihood equal to 0: (i) There is a large number of parameters to be estimated $(2 \cdot m \cdot r)$ and there is not always sufficient information, and (ii) sometimes, the parameter estimates can get close to 0 or to 1, leading to error. An additive smoothing is used for the α_c^a parameters:
$$\alpha_c^a = \frac{\gamma + \sum_{x \in \mathcal{D}} \mathbb{1}(c_x = c)\mathbb{1}(c \in L_x^a)}{2 \cdot \gamma + \sum_{x \in \mathcal{D}} \mathbb{1}(c_x = c)} \tag{3.15}$$
In this way, all possible values are reached at least once, that is, there is at least one instance of class c such that $c \in L_x^a$ and another instance of class c such that $c \in L_x^a$. In these experiments, Equation 3.15 is used instead of Equation 3.10 with $\gamma = 1$.

Datasets are simulated as follows: The ground truth class labels are distributed uniformly among all instances, that is, there are $\frac{n}{r}$ instances belonging to each class. Next, the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters are generated. In order to have annotators with different types of knowledge, a maximum ($\overline{\alpha}$) value of $\boldsymbol{\alpha}$ and a minimum value for $\boldsymbol{\beta}$ ($\underline{\beta}$) are set. All the parameters are sampled uniformly from the intervals $[0.5, \overline{\alpha}]$ and $[\underline{\beta}, 0.5]$. By means of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters, candidate sets are generated following the interpretation explained at the beginning of Section 3.4. That is, given an instance that belongs to class c, annotator a includes class c in the candidate set with probability α_c^a and each of the classes $c' \neq c$ with probability β_c^a.

Once the candidate sets are generated, 4 different schemes are used to aggregate them: (i) our EM-based method, (ii) CV (Eq. 3.3), (iii) AV and (iv) privileged aggregation (PA). The PA is obtained by computing the estimate from Eq. 3.14, using the original parameters and the ground truth class labels.

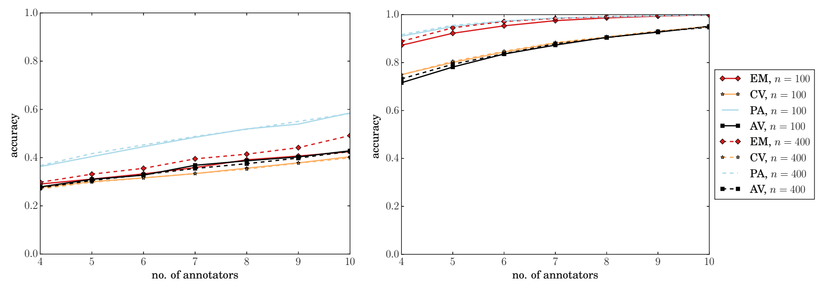
As mentioned above, EM is ensured to converge to a local maximum, so various initialisations should be carried out to achieve desirable results. In order to obtain different initialisations from the same candidate sets, we initialise the estimates $q(c|x)$ for each instance x in the following way: First, the candidate voting estimates $w_x(c)$ (Eq. 3.2) are computed for all $c \in \Omega_C$, using an additive smoothing of $\frac{1}{r}$ for each one. The $q(\cdot|x)$ are normalised so that $0 \leq q(c|x) \leq 1$ and $\sum_{c \in \Omega_C} q(c|x) = 1$. Next, to initialise $q(\cdot|x)$, a Dirichlet distribution with hyper-parameters $r \cdot w_x(c_1), \ldots, r \cdot w_x(c_r)$ is sampled: $q(\cdot|x) \sim Dir(r \cdot w_x(c_1), \ldots, r \cdot w_x(c_r))$.

30 initialisations are carried out and the values of the final $q(c|x)$ estimates that maximise the likelihood are used to infer the labels: each instance x takes the class label c that maximises $q(c|x)$. The process is repeated 30 times and the expected accuracy is approximated by computing the mean of the obtained accuracy estimates.

3.6.2 Experimental results

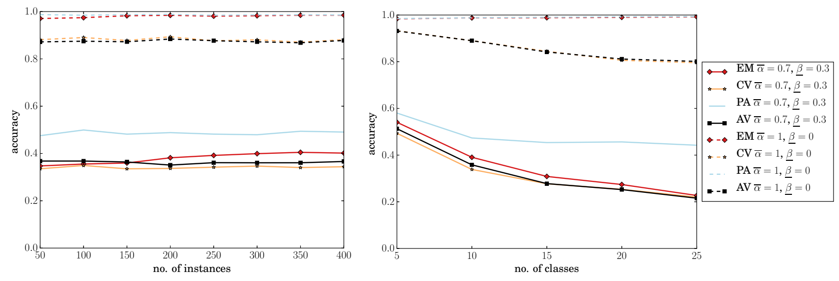
Experiments with artificial data have been performed, varying a number of parameters to compare our method and previous approaches in different scenarios.

Except for the graphics where their evolution is examined, standard values have been chosen for the parameters. The number of annotators varies from 4 to 10, although it is fixed to its standard value ($m = 7$) in different experiments. The numbers of instances used are $n = \{100, 400\}$. In the case $n = 100$,



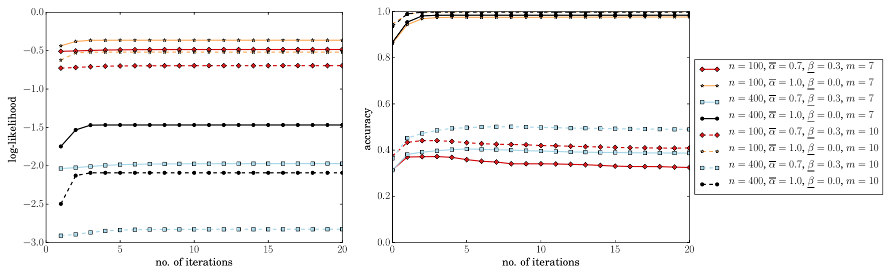
(a) $r = 10,\ \overline{\alpha} = 0.7,\ \underline{\beta} = 0.3$ (b) $r = 10,\ \overline{\alpha} = 1,\ \underline{\beta} = 0$

Fig. 3.4: Graphical description of the accuracy obtained by annotations simulated with different numbers of annotators.



(a) $m = 7,\ r = 10$ (b) $m = 7,\ n = 400$

Fig. 3.5: Graphical description of the accuracy obtained by annotations simulated with different numbers of instances and classes.



(a) Log-likelihood (scale: 10^4) (b) Accuracy

Fig. 3.6: Graphical description of the log-likelihood and the accuracy obtained throughout different iterations, with $r = 10$.

$r = \{5, 10\}$ class labels are considered, and in the case $n = 400$, $r = \{10, 20\}$ class labels are considered. Regarding the expertise of annotators, two scenarios have been studied: (i) $\underline{\beta} = 0.3$ and $\overline{\alpha} = 0.7$, where the average expertise is low, and (ii) $\underline{\beta} = 0$ and $\overline{\alpha} = 1$, where the expertise of the annotators ranges from minimum to maximum values. Due to space limitations, only the results of a representative subset of experiments are shown in this paper.

In Figures 3.4 and 3.5, the accuracy of the presented method (EM) is compared to that of the CV, the AV and the PA, in scenarios where the number of annotators (m, Fig 3.4), the number of instances (n, Fig 3.5a) and the number of classes (r, Fig 3.5b) are varied. The experimental results suggest that, in general, EM outperforms CV and AV in terms of accuracy (Eq. 1.1). The accuracies are similar only in the case where the average expertise is low and the number of classes is high with respect to the number of instances (see Figure 3.5 with $\underline{\beta} = 0.3$ and $\overline{\alpha} = 0.7$). Moreover, in the case that $\underline{\beta} = 0$ and $\overline{\alpha} = 1$ (Fig 3.4b), the proposed method reaches the accuracy of the PA.

In other words, in the presence of annotators that are experts in a subset of classes, our EM-based strategy can reach the highest possible accuracy. Note as well that the accuracy of the EM approach decreases at a smoother pace than that of CV or AV as the number of annotators is reduced.

As can be seen in Figure 3.5a, the number of instances (n) does not seem to affect the differences between the accuracies of the different methods, when it ranges between 100 and 400 (Fig 3.5a). On the other hand, the number of classes (r) has a negative effect on the accuracy of all the methods (Fig 3.5b). The only exception is that when the expertise of the annotators ranges from minimum to maximum values (Fig 3.5b, $\overline{\alpha} = 1$, $\underline{\beta} = 0$), our EM approach outperforms the baselines.

The evolution of the log-likelihood and the accuracy in each iteration of the EM can be seen in Figure 3.6. In Figure 3.6b, the accuracy in iteration number 0 is the one reached using the initial q estimates. As could be expected, generally, the log-likelihood increases monotonically and remains stable after some point (Fig 3.6a). The accuracy increases in the first iterations as well, and then remains stable in most cases (Fig 3.6b), but decreases in one case ($n = 100$, $\overline{\alpha} = 0.7$, $\underline{\beta} = 0.3$). This decline may be due to overfitting since scarce data (each annotator labels 100 instances) is used to estimate many parameters (20 per annotator).

To sum up, according to the experiments, EM seems to outperform CV and AV in most scenarios, especially when the expertise of the annotators is varied. In favourable settings, EM can reach a high accuracy - as if the real $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters were known (PA).

3.7 Conclusions

In this chapter, candidate labelling was proposed as an alternative to traditional full labelling in the context of learning from crowds. In candidate labelling, annotators provide a set of labels instead of just one. Intuitively, this simple mechanism allows for extracting more knowledge from a set of annotators. Throughout an experimental framework with artificial labels on real-world data, empirical evidence suggests that the use of candidate labelling could be profitable compared to full labelling in general terms. Moreover, it is particularly useful when (i) the number of available annotators is low, (ii) the difficulty of the instances is high, (iii) the number of possible class labels is high, or (iv) annotators are hesitant, that is, when annotators tend to doubt between several labels.

We presented an annotator reliability model that leads to an EM-based label aggregation method that extends traditional aggregation methods to the candidate labelling scenario. Experimental results obtained with artificial data suggest that our method has an enhanced performance, in terms of aggregation accuracy, compared to the baseline methods such as approval voting and candidate voting. Particularly, the difference in accuracy is larger when few annotators are available and when they show different levels of expertise.

The results obtained for label aggregation with candidate labelling are promising. An early version of this work was made available in a public repository [66] and another one was published in the *Conference of the Spanish Association of Artificial Intelligence* CAEPIA 2018 [67].

In this chapter, we have focused on label aggregation using candidate sets. However, the final objective is the learning of a classifier. In the next chapter, we propose another annotator model for candidate labelling and two classifier learning methods, one approach to sequential learning and another to joint learning. These methods extend the full labelling-based methods by Dawid-Skene [24] and Raykar et al. [13], respectively.

4

Machine learning from crowds using candidate set-based labelling

4.1 Introduction

In our way to improve the cost efficiency of crowd learning, we presented before the *candidate labelling* framework, where annotators can provide for each instance a candidate set, that is, a collection of labels, instead of a single label as in the traditional full labelling framework. We showed that it can lead to enhanced performance with a simple voting method and/or an appropriate annotator model.

In this chapter, we continue this line and propose a more complete model for the reliability of annotators and derive two different methods for learning classifiers from data labelled within this framework. Namely, we present (i) a sequential method (*SL-C*, i.e. *Sequential Learning - Candidate*) that uses an EM-based strategy to aggregate the candidate sets previous to the classifier learning step, and (ii) a joint learning method (*JL-C*, i.e. *Joint Learning - Candidate*) that performs simultaneously label aggregation and model learning. They are inspired by the works of [24] and [13], respectively, and can be seen as their generalisation from the full labelling to the candidate labelling context. This correspondence is illustrated in Table 4.1. We assume that each annotator carries out their annotation by sampling a latent scale model [68]. This model assumes that, to form the set, the inclusion of each class label is decided by independently sampling a different Bernoulli distribution. We carried out a large empirical study including the state-of-the-art approaches from the full labelling framework. The empirical results suggest that the use of techniques based on candidate labelling allows us to extract more information from the annotators than with full labelling, and hence leads to better learning results.

The structure of this chapter is as follows. Next, we propose a generative annotator model which works with candidate labels, and an EM method for sequential learning using that model. In Section 4.3, we integrate the label aggregation and the learning steps into a joint learning EM-based method. The results of an extensive set of experiments are displayed and discussed

		Learning approach	
		Sequential: aggregation before model learning	*Joint* aggregation and model learning
Type of supervision	Full labelling	Dawid-Skene [24]	Raykar et al. [13]
	Candidate labelling	SL-C, Section 4.2.2	JL-C, Section 4.3

Table 4.1: Axis of improvement faced in this chapter within the learning from crowds framework.

in Section 4.5, comparing with the state-of-the-art methods. Finally, general conclusions are drawn and possible future work is pointed out in Section 4.6.

4.2 A reliability-aware aggregation of candidate-set annotations

When aggregating the candidate sets gathered from different annotators, the contribution of each annotator can be weighted according to their reliability. In this section, we present a method that aggregates the candidate sets as the reliability of the annotators is estimated. First of all, we present the reliability model for the annotators. Then, we compute the MLE of the parameters of that model. Finally, an EM-based method is proposed for estimating the reliability parameters and aggregating the candidate sets.

4.2.1 Annotator model for candidate labelling

Let us consider that annotators produce candidate sets according to an unknown probability model over the set of labels. We assume that each annotator is dealing with an independent binary choice task for each class label. Each one consists of deciding whether to include it or not in the candidate set. This behaviour corresponds to a latent-scale model [68] that represents a probability distribution over sets of elements. A latent-scale model assumes that each element is selected independently according to a Bernoulli distribution. Due to this independence assumption, the latent choice model over sets is given as the product of the Bernoulli distributions, one for each of the available elements.

More specifically, we assume that annotators follow a latent-scale model conditioned to the true class value of the instance. Formally, let $\alpha_{ck}^{a} \in [0,1]$ denote the probability that annotator a includes label k in the candidate set for an instance with true class label c. Then, the probability of the candidate set L_x^a provided by annotator a for instance x corresponds to:

$$Pr[L_x^a] = \prod_{k\in\Omega_C} (\alpha^a_{c_x k})^{\mathbb{1}(k\in L_x^a)} \cdot (1-\alpha^a_{c_x k})^{(1-\mathbb{1}(k\in L_x^a))}$$

where Ω_C is the set of possible class labels. The annotator with complete knowledge would show $\alpha^a_{ck} = 1$ when $k = c$, and $\alpha^a_{ck} = 0$ in other cases. We denote by $\boldsymbol{\alpha} = \{\alpha^a_{ck} : c, k \in \Omega_C, a \in A\}$ the set of parameters for all the annotators. Take into account that, unlike the models of Dawid-Skene [24] or Raykar et al. [13], our α^a_{ck} parameters do not form conditional probability distributions given a fixed c. Thus, $\alpha^a_{c\cdot}$ is not a probability distribution, and $\sum_k \alpha^a_{ck}$ is not necessarily 1. An annotator could show $\alpha^a_{ck} = \alpha^a_{ck'} \approx 0.9$, meaning that they usually include both k and k' when the real label is c. This behaviour, derived from our use of candidate labelling, represents one of the main novelties of our model with respect to previous full labelling models (e.g., [13, 24]). $\boldsymbol{\alpha}$ parameters can be understood as annotator reliabilities: an annotator is more reliable as the true class has a higher probability and as the probability of the other classes is lower.

This model implicitly assumes that (i) the behaviour of an annotator only depends on the true class, $Pr[L_x^a] = Pr[L_{x'}^a]$ for $L_x^a = L_{x'}^a$ if $c_x = c_{x'}$, and (ii) for each annotator the probability of including two labels in the candidate set is conditionally independent given the actual class.

Assuming this generative model for the candidate sets and independent annotators, $Pr[\{L_x^a, L_x^{a'}\}] = Pr[L_x^a] \cdot Pr[L_x^{a'}]$, we can define the likelihood given the labelling $\mathcal{L}_x$ for instance x of real class c:

$$Pr(\mathcal{L}_x|c_x = c, \boldsymbol{\alpha}) = \prod_{a\in A}\prod_{k\in\Omega_C} (\alpha^a_{ck})^{\mathbb{1}(k\in L_x^a)} \cdot (1-\alpha^a_{ck})^{(1-\mathbb{1}(k\in L_x^a))}. \tag{4.1}$$

4.2.2 SL-C: An EM method for aggregation of candidate sets

In our first approach, we aim to learn the parameters of the described annotator model, $\boldsymbol{\alpha}$, and to aggregate the candidate sets taking it into account. That is, we want to obtain a single class label for each instance. We take advantage of the assumption that the candidate sets only depend on the real class c_x and not on the instance itself x to calculate the maximum likelihood estimates for those parameters.

Assuming that instances are i.i.d. according to $p(X, C)$, the likelihood given a dataset $\mathcal{D}$ along with a labelling $\mathcal{L}$ is:

$$Pr\left(\mathcal{L}; \boldsymbol{\alpha}\right) = \prod_{x\in\mathcal{D}}\sum_{c\in\Omega_C} Pr(\mathcal{L}_x|c_x = c; \boldsymbol{\alpha}) \cdot Pr(c_x = c). \tag{4.2}$$

By assuming that the real label c_x of each example x is unique, i.e. $Pr(c_x|x) = 1$ for all x, the marginalisation of C can be re-expressed as a product raised to the indicator function. This is a reasonable assumption as it is unlikely that two instances that have the same features x belong to different classes. Then, using Eq. 4.1 we have that the likelihood can be rewritten as:

$$\begin{aligned} Pr\left(\mathcal{L};\boldsymbol{\alpha}\right) &= \prod_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}} \left[Pr(\mathcal{L}_x|c_x=c;\boldsymbol{\alpha})\right]^{\mathbb{1}(c_x=c)} \\ &= \prod_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}} \Big[\prod_{\substack{a\in A\\ k\in\Omega_C}} (\alpha^a_{ck})^{\mathbb{1}(k\in L^a_x)}\cdot(1-\alpha^a_{ck})^{(1-\mathbb{1}(k\in L^a_x))}\Big]^{\mathbb{1}(c_x=c)}. \end{aligned} \tag{4.3}$$

The values of α^a_{ck} that maximise this expression are the same as the ones that maximise the log-likelihood. The log-likelihood is:

$$\begin{aligned} \log\left[Pr\left(\mathcal{L};\boldsymbol{\alpha}\right)\right] = \sum_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}} \mathbb{1}(c_x=c)\cdot\Big[\sum_{\substack{a\in A\\ k\in\Omega_C}} \mathbb{1}(k\in L^a_x)\cdot\log(\alpha^a_{ck}) \\ +(1-\mathbb{1}(k\in L^a_x))\cdot\log(1-\alpha^a_{ck})\Big]. \end{aligned} \tag{4.4}$$

Let us compute the derivative of the log-likelihood with respect to $\alpha^{a'}_{c'k'}$, for specific values $a'\in A$ and $c',k'\in\Omega_C$, and equal it to 0 to find its maximum:

$$\frac{d\log\left[Pr\left(\mathcal{L};\boldsymbol{\alpha}\right)\right]}{d\alpha^{a'}_{c'k'}} = \sum_{x\in\mathcal{D}} \mathbb{1}(c_x=c')\cdot\left(\frac{\mathbb{1}(k'\in L^{a'}_x)}{\alpha^{a'}_{c'k'}} - \frac{1-\mathbb{1}(k\in L^a_x)}{1-\alpha^a_{ck})}\right)=0. \tag{4.5}$$

Solving this expression we obtain the MLE of α^a_{ck}:

$$\alpha^a_{ck} = \frac{\sum_{x\in\mathcal{D}}\mathbb{1}(c_x=c)\cdot\mathbb{1}(k\in L^a_x)}{\sum_{x\in\mathcal{D}}\mathbb{1}(c_x=c)}, \tag{4.6}$$

which is the proportion of instances x of real class $c_x=c$ for which annotator a included label k in their candidate set, $k\in L^a_x$. Note that for computing Equation 4.6, we need to know the true class labels c_x of all the instances, which is clearly unrealistic in the crowd learning framework.

We could substitute the indicator functions by an estimation of the ground truth labels in Equation 4.6. In our setting, an expression for such an estimation of the probability that instance x belongs to class label c given the candidate sets can be obtained as $Pr\left(c_x=c|\mathcal{L}_x;\boldsymbol{\alpha}\right)$. Using the Bayes rule and Eq. 4.1, we estimate the probability of the true class as:

$$\begin{aligned} Pr\left(c_x=c|\mathcal{L}_x;\boldsymbol{\alpha}\right) &\propto Pr(c_x=c)\cdot Pr\left(\mathcal{L}_x|c_x=c;\boldsymbol{\alpha}\right) \\ &\propto Pr(c_x=c)\cdot\prod_{\substack{a\in A\\ k\in\Omega_C}} (\alpha^a_{ck})^{\mathbb{1}(k\in L^a_x)}\cdot(1-\alpha^a_{ck})^{(1-\mathbb{1}(k\in L^a_x))}, \end{aligned} \tag{4.7}$$

where $Pr(c_x=c)$ is calculated as the relative frequency of label c. Note that the rest of the class labels ($k\in\Omega_C : k\neq c$) intervene through the use of the $\boldsymbol{\alpha}$ parameters: it accounts for the usual confusions of the annotators; i.e., the probability that an annotator introduces a wrong label k when the real

Algorithm 1 Method *SL-C*

```
1: **Input**:
2: $\mathcal{L}$: Crowdsourced labelling
3: $\boldsymbol{\alpha}^{(0)}$: Initial guess of the parameters
4: **Output**:
5: $q_{\boldsymbol{\alpha}}^{(t)}$: Probability distribution
6: $\boldsymbol{\alpha}^{(t)}$: Final parameters
7: procedure EM-SL-C
8:   $t \leftarrow 0$
9:   while $t = 0$ or $\boldsymbol{\alpha}^{(t)} \neq \boldsymbol{\alpha}^{(t-1)}$ do
10:    $t \leftarrow t + 1$
11:    E-step:
12:    for $c \in \Omega_C$ and $x \in \mathcal{D}$ do
13:      $q_{\boldsymbol{\alpha}}^{(t)}(c|x) \leftarrow Pr\left(c|\mathcal{L}_x; \boldsymbol{\alpha}^{(t-1)}\right)$
14:    M-step:
15:    $\boldsymbol{\alpha}^{(t)} \leftarrow \arg\max_{\boldsymbol{\alpha}} \mathbb{E}_{\boldsymbol{c}\sim q_{\boldsymbol{\alpha}}^{(t)}} \log Pr(\mathcal{L}|\boldsymbol{c}; \boldsymbol{\alpha})$
16:   return $q_{\boldsymbol{\alpha}}^{(t)}, \boldsymbol{\alpha}^{(t)}$
```

one is c. Note that for computing Equation 4.7, the model parameters $\boldsymbol{\alpha}$ are required.

The reader will have noticed the mutual requirements of Equations 4.6 and 4.7: to estimate the ground truth labels we need the model parameters $\boldsymbol{\alpha}$ and to calculate $\boldsymbol{\alpha}$ we need an estimation of the ground truth labels. This naturally leads to an EM method that iterates over two steps: (i) E-step, where the expected value of the ground truth label of every instance x is obtained with Equation 4.7 (given the current $\boldsymbol{\alpha}$ parameters), and (ii) M-step, where the annotator reliability parameters $\boldsymbol{\alpha}$ are updated with Equation 4.6 (given the ground truth estimations of the previous E-step). If we define the computation of E-Step as:

$$q_{\boldsymbol{\alpha}}(c|x) = Pr\left(c_x = c|\mathcal{L}_x; \boldsymbol{\alpha}\right),$$

these $q_{\boldsymbol{\alpha}}(c|x)$ estimates can substitute the indicator function $\mathbb{1}(c_x = c)$ in Equation 4.6, accounting for all the possible values of c_x probabilistically, since the real class label is unknown:

$$\alpha_{ck}^{a} = \frac{\sum_{x\in\mathcal{D}} q_{\boldsymbol{\alpha}}(c|x) \cdot \mathbb{1}(k \in L_x^a)}{\sum_{x\in\mathcal{D}} q_{\boldsymbol{\alpha}}(c|x)}. \tag{4.8}$$

Algorithm 1 describes our EM-based method named **SL-C**, which stands for sequential learning with candidate labelling. The complexity of the initialisation is $\mathcal{O}(nmr^2)$ with respect to the numbers of instances (n), annotators (m) and classes (r). For each iteration, the complexity of both the E-step (line 13 in Algorithm 1) and the M-step (line 15 in Algorithm 1) is also $\mathcal{O}(nmr^2)$. Thus, the overall complexity of each iteration is $\mathcal{O}(nmr^2)$.

The E and M-steps are iteratively interleaved until convergence. In our implementation, convergence is reached when the difference between the estimated parameters in two consecutive iterations falls below a threshold. The likelihood is enhanced in each EM iteration until a local maximum is reached at convergence [17]. Thus, our algorithm stops when the MLE $\boldsymbol{\alpha}$ cannot be further improved. The result of SL-C is an estimate of the ground truth, along with estimates for the model parameters $\boldsymbol{\alpha}$. To complete the goal of learning from crowds, a classifier should be learned posteriorly using standard supervised classification techniques (sequential approach).

This algorithm can be seen as an extension of the Dawid-Skene method [24] to the scenarios where annotators provide candidate sets instead of single labels. We will use that method, briefly explained in Section 1.3, as a baseline method to compare with, as it is the equivalent to our method SL-C in the full labelling context.

4.3 Reliability-aware joint aggregation and learning from candidate sets

In the previous section, we presented the method SL-C (Alg. 1), which deals with the aggregation of candidate sets contributed by different annotators under some strong assumptions. The method SL-C is sequential, meaning that once it aggregates the candidate sets, we need a second stage where a classifier is learned from the ground truth estimations, $q_{\boldsymbol{\alpha}}(c|x)$. Our second proposal performs the classification model learning and the aggregation in a single step (joint learning) and makes use of the descriptive feature in that process. Thus, we are able to learn a classification model and a reliability model for the annotators at the same time.

Let us assume again the annotator reliability model described in Section 4.2.1. Let us also consider a probabilistic classifier which models the probability that instance x belongs to class c, expressed as $h(c|x;\theta)$. Here, θ represents the classification model parameters. The method presented in this section has, as the main goal, to train classifier h such that it is able to predict the class label of unseen examples.

Under the assumption that all instances are i.i.d. according to $p(X,C)$ and that annotators provide their candidate sets independently and following the generative model described above, the likelihood is given by

$$\begin{aligned}
Pr\left(\mathcal{D},\mathcal{L}|\boldsymbol{\alpha},\theta\right) &= \prod_{x\in\mathcal{D}}\sum_{c\in\Omega_C} Pr\left(\mathcal{L}_x|c_x=c,\boldsymbol{\alpha}\right) Pr\left(c_x=c|x,\theta\right)\\
&= \prod_{x\in\mathcal{D}}\sum_{c\in\Omega_C} Pr\left(\mathcal{L}_x|c_x=c,\boldsymbol{\alpha}\right) h(c|x;\theta)\\
&= \prod_{x\in\mathcal{D}}\sum_{c\in\Omega_C}\left[h(c|x;\theta)\prod_{\substack{a\in A\\ k\in\Omega_C}}(\alpha^a_{ck})^{\mathbb{1}(k\in L^a_x)}\cdot(1-\alpha^a_{ck})^{(1-\mathbb{1}(k\in L^a_x))}\right].
\end{aligned} \tag{4.9}$$

Assuming that every example $x\in\mathcal{D}$ has a single ground truth label c_x (that is, $Pr(c_x|x)=1$), the marginalisation of C (sum over $c\in\Omega_C$) in Equation 4.9 can be represented as a product of factors to the power of the indicator function on the actual value of the class variable, as we did for Equation 4.3. Thus, the likelihood could be rewritten as:

$$Pr\left(\mathcal{D},\mathcal{L}|\boldsymbol{\alpha},\theta\right) = \prod_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}}\left[h(c|x;\theta)\cdot\prod_{\substack{a\in A\\ k\in\Omega_C}}(\alpha^a_{ck})^{\mathbb{1}(k\in L^a_x)}(1-\alpha^a_{ck})^{(1-\mathbb{1}(k\in L^a_x))}\right]^{\mathbb{1}(c_x=c)}, \tag{4.10}$$

and the log-likelihood is

$$\begin{aligned}
\log\left(Pr\left(\mathcal{D},\mathcal{L}|\boldsymbol{\alpha},\theta\right)\right) = \sum_{\substack{x\in\mathcal{D}\\ c\in\Omega}}\mathbb{1}(c_x=c)\cdot\Big[\log\left(h(c|x;\theta)\right)+\\
+\sum_{\substack{a\in A\\ k\in\Omega}}\Big(\mathbb{1}(k\in L^a_x)\log(\alpha^a_{ck})+(1-\mathbb{1}(k\in L^a_x))\log(1-\alpha^a_{ck})\Big)\Big].
\end{aligned} \tag{4.11}$$

Given this expression, we can obtain the MLE for the parameters α^a_{ck} by computing the derivative of the log-likelihood with respect to α^a_{ck}, and finding the value of the parameter when the derivative is equal to zero. Noting that $h(c|x;\theta)$ does not directly depend on α^a_{ck} in Eq. 4.11, the MLE for α^a_{ck} turns out to have exactly the same expression of Equation 4.6. Thus, we face again the need of the real class labels c_x for estimating the $\boldsymbol{\alpha}$ parameters, and also to learn the classifier h. However, as aforementioned, this piece of information is missing in learning from crowds. Thus, we resort again to an EM method to obtain the MLE for our model parameters $(\boldsymbol{\alpha},\theta)$.

4.3.1 JL-C: An EM method for jointly aggregating candidate sets and learning

In the E-step, we estimate the probability of label $c\in\Omega_C$ for instance $x\in\mathcal{D}$ given the parameter estimates $\boldsymbol{\alpha}$ and θ making use of the Bayes rule as:

$$Pr(c_x=c|\mathcal{L}_x,x;\boldsymbol{\alpha},\theta) = \frac{Pr(\mathcal{L}_x|c_x=c;\boldsymbol{\alpha},\theta)\cdot Pr(c_x=c|x;\boldsymbol{\alpha},\theta)}{Pr(\mathcal{L}_x;\boldsymbol{\alpha},\theta)}, \tag{4.12}$$

where $Pr(c_x = c|x; \boldsymbol{\alpha}, \theta) = h(c|x;\theta)$ is given by the classifier h with parameters θ. We denote:

$$q_{\boldsymbol{\alpha},\theta}(c|x) = Pr\left(c_x = c|\mathcal{L}_x, x; \boldsymbol{\alpha}, \theta\right),$$

the probabilistic estimate of the ground truth of x, which can be re-expressed as:

$$q_{\boldsymbol{\alpha},\theta}(c|x) \propto h(c|x;\theta) \prod_{\substack{a \in A \\ k \in \Omega_C}} (\alpha^a_{ck})^{\mathbb{1}(k \in L^a_x)} \cdot (1 - \alpha^a_{ck})^{(1-\mathbb{1}(k \in L^a_x))}, \tag{4.13}$$

where $\boldsymbol{\alpha}$ and θ are the parameter estimates found in the previous EM iteration. As before (in Eq. 4.7, for SL-C), the probability estimate $q_{\boldsymbol{\alpha},\theta}(c|x)$ depends on all the labels other than c through the $\boldsymbol{\alpha}$ parameters, and it is also proportional to the probability predicted by classifier h. Note that, in this case, x is taken into account through the classifier to soften the assumption of the annotator model that the behaviour of the annotators only depends on the real label.

The M-step uses the distributions $q_{\boldsymbol{\alpha},\theta}(c|x)$ to fit the model parameters $\boldsymbol{\alpha}$ and θ. As before, since the real labels are missing, we find the $\boldsymbol{\alpha}$ estimates that maximise the expectation $\mathbb{E}_{\boldsymbol{c} \sim q_{\boldsymbol{\alpha},\theta}} \log Pr(\mathcal{L}|\boldsymbol{c}; \boldsymbol{\alpha})$, which implies substituting the indicator functions in Eq. 4.6 with $q_{\boldsymbol{\alpha},\theta}(c|x)$, as follows:

$$\alpha^a_{ck} = \frac{\sum_{x \in \mathcal{D}} q_{\boldsymbol{\alpha},\theta}(c|x) \cdot \mathbb{1}(k \in L^a_x)}{\sum_{x \in \mathcal{D}} q_{\boldsymbol{\alpha},\theta}(c|x)}. \tag{4.14}$$

The $q_{\boldsymbol{\alpha},\theta}(c|x)$ estimates are also used for learning the classification model parameters θ using a training dataset with probabilistic labelling: the pair $(x, c) \in (\mathcal{D}, \Omega_C)$ has weight $q_{\boldsymbol{\alpha},\theta}(c|x)$. The new fit of θ would be just the parametrisation of classifier h trained using this weighted training sample.

Algorithm 2 describes the method named **JL-C**, which stands for joint learning with candidate labelling. The computational complexity of the initialisation is $\mathcal{O}(f + nmr^2)$, where f represents the complexity of fitting the chosen classifier h. For each iteration, the complexity of the E-step (line 14 in Algorithm 2) is $\mathcal{O}(gnmr^2)$, where g represents the complexity of the prediction using the chosen classifier h. The complexity of the M-step (lines 17 and 16 in Algorithm 2) is $\mathcal{O}(f + nmr^2)$. Thus, the overall complexity of each iteration is $\mathcal{O}(f + gnmr^2)$.

This method can be seen as an extension of the one by Raykar et al. [13] to the candidate labelling scenario. Their joint learning technique is used in this chapter as a baseline for our method, as it is equivalent to JL-C in the full labelling framework.

The rest of the details of the JL-C method are implemented in the same way as for SL-C.

Algorithm 2 Method *JL-C*

```
1: Input:
2: $\mathcal{D}$: Training set
3: $\mathcal{L}$: Crowdsourced labelling
4: $(\boldsymbol{\alpha}^{(0)}, \theta^{(0)})$: Initial guess of the parameters
5: Output:
6: $q^{(t)}_{\boldsymbol{\alpha},\theta}$: Probability distribution
7: $(\boldsymbol{\alpha}^{(t)}, \theta^{(t)})$: Final parameters
8: procedure EM-JL-C
9:   $t \leftarrow 0$
10:  while $t = 0$ or $\boldsymbol{\alpha}^{(t)} \neq \boldsymbol{\alpha}^{(t-1)}$ do
11:    $t \leftarrow t + 1$
12:    E-step:
13:    for $c \in \Omega_C$ and $x \in \mathcal{D}$ do
14:      $q^{(t)}_{\boldsymbol{\alpha},\theta}(c|x) \leftarrow Pr\left(\boldsymbol{c}|\mathcal{L}, \mathcal{D}; \boldsymbol{\alpha}^{(t-1)}, \theta^{(t-1)}\right)$
15:    M-step:
16:    $\boldsymbol{\alpha}^{(t)} \leftarrow \arg\max_{\boldsymbol{\alpha}} \mathbb{E}_{\boldsymbol{c} \sim q^{(t)}_{\boldsymbol{\alpha},\theta}} \log Pr(\mathcal{L}|\boldsymbol{c}; \boldsymbol{\alpha})$
17:    $\theta^{(t)} \leftarrow \arg\max_{\theta} \mathbb{E}_{\boldsymbol{c} \sim q^{(t)}_{\boldsymbol{\alpha},\theta}} \log Pr(\boldsymbol{c}|\mathcal{D}; \theta)$
18:  return $q^{(t)}_{\boldsymbol{\alpha},\theta}, \boldsymbol{\alpha}^{(t)}, \theta^{(t)}$
```

4.4 Model selection and initialisation

As aforementioned, the EM algorithm is guaranteed to converge to a local optimum. The quality of this local optimum depends on the initial values from which the method departs, so practitioners are usually advised to run any EM-based algorithm multiple times to try to reach different local maxima and keep only the best model. The criterion to select the best model is usually maximum likelihood.

For SL-C, we need to set initial values for $\boldsymbol{\alpha}$ and for JL-C we need to fill in θ too. Those values could be generated randomly, but it is reasonable to consider the available information also for this step. In this way, we can initialise the method by learning a first fit of the model ($\boldsymbol{\alpha}^{(0)}$ in the case of SL-C, $(\boldsymbol{\alpha}^{(0)}, \theta^{(0)})$ in the case of JL-C) from a dataset completely labelled. Candidate voting (Eq. 3.2) could be used to obtain such a complete dataset.

As the estimation of $\boldsymbol{\alpha}$ is deterministic, if so is the estimation of θ (which depends on the type of classifier), the whole EM procedure is also deterministic given a certain initialisation. In this way, we introduce a random component in the initial label aggregation (together with Eq. 3.2) to avoid the otherwise deterministic behaviour of our EM method and thus try to reach different local maxima. We use random initialisations as follows: First, (i) obtain the candidate voting estimate as in Equation 3.2 for every $(x, c) \in (\mathcal{D}, \Omega_C)$. Then, (ii) sample the probability distributions provided by the candidate voting estimates for each $x \in \mathcal{D}$ to obtain initial deterministic guesses. And (iii) use these guesses as the ground truth estimates to compute the initial fit of $\boldsymbol{\alpha}$ using Eq. 4.6. For JL-C, we also use the initial guesses as labelling for training the classification model parameters θ. The proposed randomised initialisation preserves the contribution of the annotators and incorporates information from the descriptive features.

After running the EM method several times with different initialisations, several models with different fits of the parameters are obtained. The model that maximises the expected log-likelihood is kept. For SL-C, we calculate the expected log-likelihood by applying the $q_{\boldsymbol{\alpha}}(c|x)$ estimates computed in the last E-step to Equation 4.3:

$$E_{q_{\boldsymbol{\alpha}}}\left[\log\left(Pr\left(\mathcal{L}|\boldsymbol{\alpha}\right)\right)\right] = \sum_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}} q_{\boldsymbol{\alpha}}(c|x) \cdot \sum_{\substack{a\in A\\ k\in\Omega_C}} \Big(\mathbb{1}(k \in L_x^a) \cdot \log(\alpha_{ck}^a) + (1 - \mathbb{1}(k \in L_x^a)) \cdot \log(1 - \alpha_{ck}^a)\Big). \tag{4.15}$$

Analogously, for JL-C, we use the corresponding $q_{\boldsymbol{\alpha},\theta}$ estimates for calculating the expected log-likelihood:

$$E_{q_{\boldsymbol{\alpha},\theta}}\left[\log\left(Pr\left(\mathcal{D},\mathcal{L}|\boldsymbol{\alpha},\theta\right)\right)\right] = \sum_{\substack{x\in\mathcal{D}\\ c\in\Omega_C}} q_{\boldsymbol{\alpha},\theta}(c|x) \cdot \Big[\log\left(h(c|x;\theta)\right) + \sum_{\substack{a\in A\\ k\in\Omega_C}} \Big(\mathbb{1}(k \in L_x^a)\log(\alpha_{ck}^a) + (1 - \mathbb{1}(k \in L_x^a))\log(1 - \alpha_{ck}^a)\Big)\Big]. \tag{4.16}$$

4.5 Experiments

We have carried out an empirical analysis of both presented methods: SL-C and JL-C. The main hypothesis in this chapter is that candidate labelling provides more information about the true classes than the classical full labelling, which can lead to classifiers with better performance. To check this hypothesis, we test the performance of our methods against that of Dawid-Skene [24] (DS), as the sequential approach analogous to SL-C in the full labelling context, and that of Raykar et al. [13] (RAY), as the joint learning method analogous to JL-C in full labelling.

Besides, the experiments are designed to analyse relative differences in behaviour between SL-C and JL-C, as a way to compare the sequential and joint learning approaches. Unfortunately, there are no real crowdsourced datasets that make use of candidate labelling, so we have resorted to generating synthetic data, which allows us to explore a wider range of experimental scenarios.

4.5.1 Synthetic label generation

Crowdsourced labels are simulated departing from standard supervised data and weakly supervised data with partial labels (PL) [15]. The latter kind of data has been chosen as candidate labelling is analogous to that in the learning from crowds framework. A general procedure that allows generating both full and candidate crowdsourced labels synthetically is used.

The synthetic label generation procedure for standard supervised datasets is as follows. We have a set of m annotators A, and each annotator $a \in A$ is simulated by means of a set of probability distributions with support in Ω_C, $\{g_a(\cdot|c)\}_{c \in \Omega_C}$. The annotators are sampled from a Dirichlet distribution with r hyperparameters, all equal to 1 except for the c-th one, which is equal to $\beta \geq 1$. This experimental parameter allows us to control the expertise of the annotator: the greater the β value is, the higher tends to be the probability of the c-th class. Note that when $\beta = 1$ annotators are generated such that, on average, their labels are uniformly selected, and thus they do not provide useful information about the true class.

Given an instance $x \in \mathcal{D}$ with an associated class label $c_x \in \Omega_C$, the labelling is generated by sampling the probability distribution $g_a(\cdot|c_x)$. That probability distribution is sampled once to perform **full labelling** (annotators express no doubt), or $\lceil prop \cdot r \rceil$ **times with replacement** to perform **candidate labelling** (annotators provide multiple labels to express their doubts). In the case that $prop \leq 1/r$, there will be only one label in the candidate set and it will be equivalent to full labelling, and when $prop > 1/r$, the size of the candidate set is in $[1, \lceil prop \cdot r \rceil]$.

When we use weakly supervised data with partial labels, where each instance $x \in \mathcal{D}$ is associated with a partial label set $C_x \subseteq \Omega_C$, the procedure is the same as above with a single exception. Instead of $g_a(\cdot|c_x)$, we sample the probability distribution $\overline{g_a}(\cdot|x) = \sum_{c \in \mathcal{C}_x}(g_a(\cdot|c)/|\mathcal{C}_x|$ to generate the labelling for instance $x \in \mathcal{D}$.

As aforementioned, for the sake of a fair comparison, the same annotator model is used for generating full and candidate labelling, which only differs in the value of *prop*. The inevitable difference appears when a set of labels needs to be generated to build the candidate set. We also would like to highlight that this generative model is more complex than the models underlying SL-C, JL-C, DS and RAY. Thus, none of them is in an advantageous position in the following experimental design with that respect.

4.5.2 Experimental design

This empirical study covers a wide range of experimental scenarios by using different configurations of the data generative process. We use 6 fully labelled datasets from the UCI repository (http://archive.ics.uci.edu/ml) and 3 partially labelled datasets. The selected datasets and their characteristics are listed in Table 4.2. We simulate different numbers of annotators $m \in \{3, 5, 7, 9\}$, and different degrees of expertise for them $\beta \in \{1, 3, 5, 7\}$. For candidate labelling generation, the proportion of sampled labels takes values $prop \in \{0.1, 0.3, 0.5, 0.7\}$. We have used two classifiers from very differ-

Name	n	d	r
Dermatology	366	34	6
Glass	214	9	6
Segment	2310	19	7
Svmguide4	612	10	6
Vehicle	846	18	4
Vowel	990	10	11
Birdac	3718	38	13
Lost	1122	108	14
MSRCv2	1758	48	23

Table 4.2: Selected supervised datasets from UCI repository [47] and partially labelled datasets. The columns display, in the following order: Name of the dataset, number of instances (n), dimension of the explanatory variable (d) and number of classes (r).

ent families from *sklearn 0.22.1* with default parameters: 5-Nearest Neighbour (5NN) and Random Forest (RF).

Model performance is measured as the area under the receiver operating characteristic curve (AUC). As ours are all multi-class classification problems ($r > 2$), AUC is computed as the average value after a *one-vs-rest* strategy. It is estimated using stratified 5-fold cross-validation, where the test sets are fully supervised. To reduce the issue that the learning methods get trapped in the same local optima always, all of them are always run twice, and the output model with the highest expected log-likelihood is kept. Expected log-likelihood is measured by Eq. 4.16 for JL-C and Raykar et al. [13] (as full labelling is just a particular case of candidate labelling), and by Eq. 4.15 for SL-C.

4.5.3 Results

Figure 4.1 shows the impact of the expertise (β parameter) of the annotators on the performance of the methods. It displays the results, in terms of the AUC metric, of different experiments where increasing values for parameter β are used (the rest of the parameters are fixed to default intermediate values[1]). As expected and observed in Section 3.3, all methods consistently show better performance as the expertise of the annotators increases. As the annotator expertise decreases and gets closer to $\beta = 3$ (the scenario that is expected to be closest to reality, as annotators are non-expert), the performance difference of our methods relative to that of RAY and DS tends to become larger.

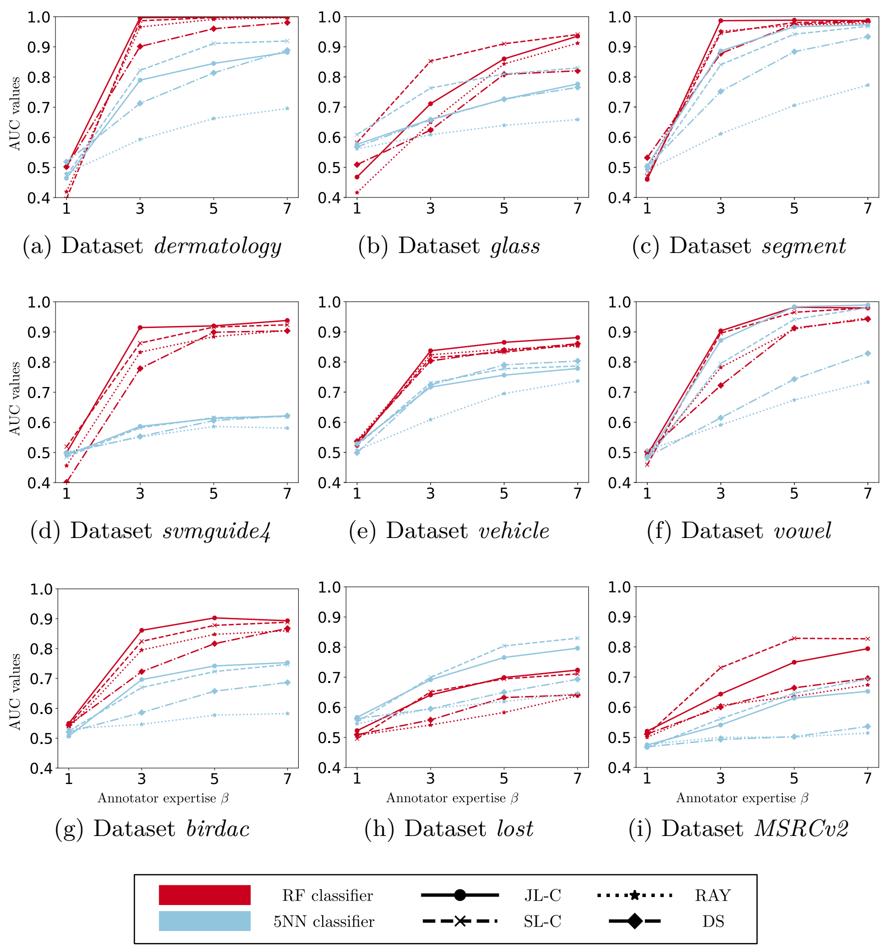
[1] Fixed parameters use values $m = 5$, $\beta = 3$ and $prop = 0.5$, selected as they draw the elbow of the curves of the respective parameter performance curves in most of the cases.

When $\beta = 1$, the AUC scores are always near 0.5 (virtually, random classifiers). This is coherent with the fact that, with $\beta = 1$, annotators provide random labels without any information about the true class. Note that this is not the usual case, but gives us a reference to compare with. Overall, JL-C and SL-C outperform RAY and DS. There are cases where RAY or DS are competitive regarding our methods, usually when there is little growth in the AUC score from $\beta = 3$ values on. This might be due to limited problem difficulty, as little information on supervision leads to the best performance that the specific classifier type can reach.

Figure 4.2 shows the impact of the number of annotators (m) on the performance of the methods. As expected, the performance tends to improve as the number of annotators increases. The steepest performance increases are most commonly observed between $m = 3$ and $m = 5$ (clearly with JL-C and 5NN), and less commonly between $m = 5$ and $m = 7$. Usually, the degree of improvement of RAY and DS as the number of annotators increases is smaller than that of SL-C and JL-C. With fewer annotators, which is a realistic setup, our methods have an advantage in most scenarios, except for JL-C with 5NN. The performance gain is barely observed in datasets for which the AUC is close to 1 for $m = 3$ (*dermatology*, and *segment*).

Figure 4.3 shows the effect of the maximum candidate set size (*prop*) on the performance. Although RAY and DS are not affected by this parameter, they are included for the sake of comparability. Overall, as *prop* increases (flexibility of annotators increases) the results get better.When *prop* = 0.1 (non-flexible annotators), most of the time annotators provide a single label for each instance, as in full labelling. In that case, our models for SL-C and JL-C become virtually equivalent to those of DS and RAY, respectively. Thus, the results of SL-C and DS are similar, as well as those of JL-C and RAY (random labelling generation might explain occasional small divergences). In general, as the value of *prop* gets larger (flexibility of annotators increases), the performance of our methods improves with respect to the baselines. These results, which are in line with the ones observed in Section 3.3, indicate that annotators should be encouraged to provide candidate sets large enough to ensure that they contain the real class label. In some cases, providing too many labels (*prop* = 0.7) could also lead to poorer results, although they would still perform better than the baselines.

To assess significant differences for each data set and each parameter configuration, we have performed a two-sample t-test with $\alpha = 0.05$ to compare the four methods pairwise. SL-C significantly outperforms DS in 58.54% of the configurations, while the opposite never happens (when comparing against RAY, SL-C has a better performance in 67.08% of the configurations, and the opposite occurs in only 0.83%). JL-C performs significantly better than RAY in 40% of configurations while RAY never obtains a significant advantage (it significantly outperforms DS in 50.41% of configurations, and the opposite never happens). JL-C outperforms SL-C in 17.45% of configurations, and SL-C outperforms JL-C in 20.57%. Additional figures with alternative datasets



(a) Dataset *dermatology* (b) Dataset *glass* (c) Dataset *segment*

(d) Dataset *svmguide4* (e) Dataset *vehicle* (f) Dataset *vowel*

(g) Dataset *birdac* (h) Dataset *lost* (i) Dataset *MSRCv2*

Fig. 4.1: Experimental results throughout different values of the parameter β (annotator expertise), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $m = 5$ and $prop = 0.5$.

and configurations are available in Appendix B. Similar behaviours to those displayed here are observed.

To sum up, the presented methods (SL-C and JL-C) outperform the baselines (RAY and DS) in most of the configurations, in terms of AUC (see Table 4.3). Their performance is enhanced as the number of annotators and level of expertise are increased (differently depending on the classifier used).

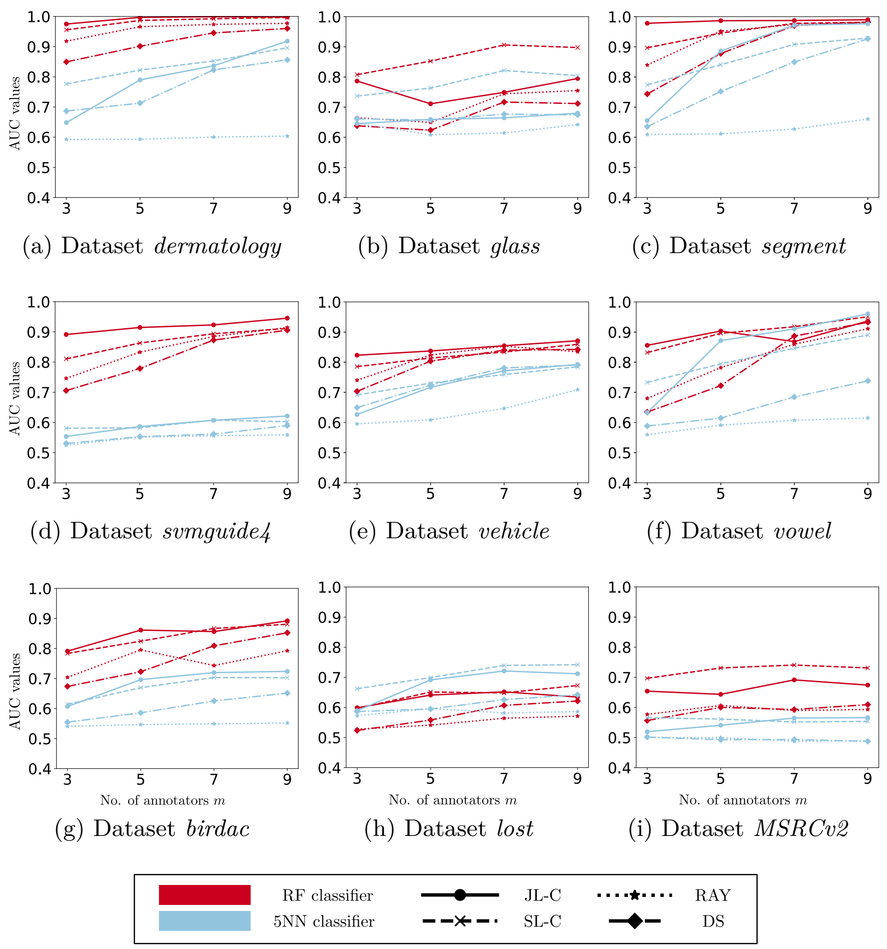
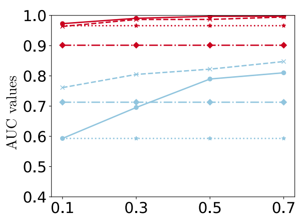
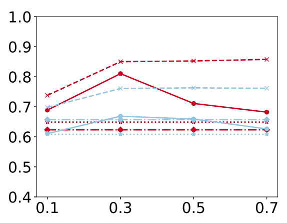


Fig. 4.2: Experimental results throughout different values of the parameter m (number of annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 3$ and $prop = 0.5$.

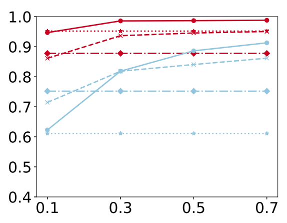
In general, by allowing annotators to provide more classes (*prop*), both SL-C and JL-C show a performance improvement. Between SL-C and JL-C, it seems they mutually outperform each other depending heavily on the classifier and the dataset, with virtually no preference among them.



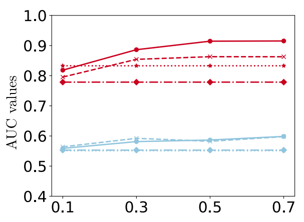
(a) Dataset *dermatology*



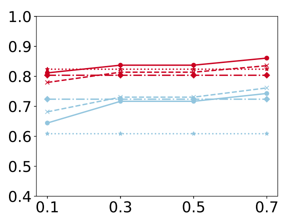
(b) Dataset *glass*



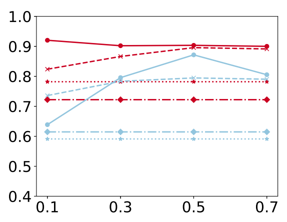
(c) Dataset *segment*



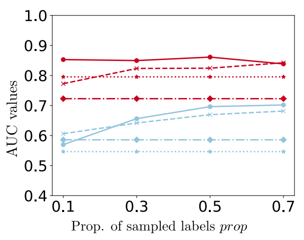
(d) Dataset *svmguide4*



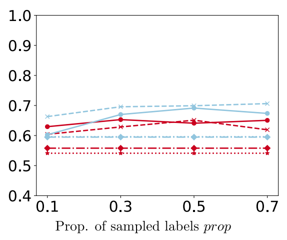
(e) Dataset *vehicle*



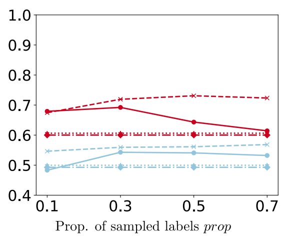
(f) Dataset *vowel*



(g) Dataset *birdac*



(h) Dataset *lost*



(i) Dataset *MSRCv2*

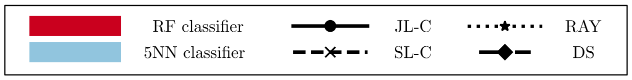


Fig. 4.3: Experimental results throughout different values of the parameter *prop* (flexibility of the annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 3$ and $m = 5$.

4.5.4 Discussion

Based on this empirical study, we can put forward several ideas.

Candidate labels seem to gather more discriminative information than the classic full labelling: SL-C and JL-C outperform RAY and DS in a vast majority of experimental scenarios. Using candidate labels we can produce

	A	TIE	B
SL-C (A) vs. DS (B)	58.54%	41.46%	0%
SL-C (A) vs. RAY (B)	67.08%	32.09%	0.83%
JL-C (A) vs. RAY (B)	40%	60%	0%
JL-C (A) vs. DS (B)	50.41%	49.59%	0%
JL-C (A) vs. SL-C (B)	17.45%	61.98%	20.57%

Table 4.3: General experimental results. The percentages of the configurations where one method obtains significantly higher AUC values than the other or there is a tie, according to the t-test, are displayed.

classifiers with at least equal performance than using full labelling, with fewer annotators or with lower-expertise annotators. Remember that the most basic objective of crowdsourced labelling is to reduce the cost of obtaining supervised data. Empirical evidence gathered in this section shows that using candidate labelling would be a way to further reduce the cost of data labelling compared to classical full labelling.

The ability to express doubts about the labelling provides extra information about the true class. The two presented methods consistently improve with annotators that on average provide a larger number of labels (Figure 4.3). This evidence should at first motivate practitioners to allow annotators to provide sets of labels and encourage them to be as flexible as needed. A fair instruction would be to indicate to annotators that including the correct answer in the candidate set is preferred rather than filtering incorrect answers out. However, we need to be careful with these instructions: a set with too many labels might become uninformative and reduce the performance of the methods.

Other features used in the design of the empirical study do not show any light on the comparison between methods or labelling approaches. Sometimes, the same performance is reached by our candidate labelling-based methods with smaller feature values. However, all of them show the already-known trend of enhancement.

The computational complexity of our methods is similar to that of the baselines they were inspired by. We tested the **scalability** of our methods and the results suggest that, with an increasing number of instances, the running time of SL-C is always similar to that of the employed classifier, meanwhile for JL-C it seems to grow exponentially when employing RF and linearly in the case of 5NN. The variable that has the greatest effect on the running time seems to be the number of classes while increasing the number of annotators seems to cause a small increase. The figures that graphically summarise the scalability test are available in Appendix C.

Sequential or joint learning. Arguably any crowd learning method could be categorised as (i) methods that first estimate the ground truth and then use standard machine learning to learn from it, and (ii) methods that learn a model as the ground truth labels are estimated. We presented, for candidate labelling, a method from each category. Our empirical study does not show relevant performance differences between them (see Table 4.3). This suggests that practitioners should test both approaches and empirically select the most appropriate one for their problems.

Finally, the **annotator model** for candidate labelling is one of our contributions (see Section 4.2.1). Both proposed methods use it, and their enhanced performance regarding that of DS and RAY validates it. Nevertheless, these methods could be easily adapted to work with other annotator models. Similarly, this empirical study is influenced by the type of classifier learned (5NN and RF). Nevertheless, our methods are completely abstracted from the classifier type and could work with any probabilistic classifier.

4.6 Conclusions

In this chapter, we presented two methods for learning classifiers from data labelled by annotators that provide sets of candidate labels (instead of standard full labelling, that is, providing single labels). We propose an annotator model and two methods which can be seen as extensions of two state-of-the-art works from full labelling to the candidate labelling framework. Both proposed approaches deal with the aggregation and model learning steps sequentially and jointly, respectively.

The extensive empirical analysis performed in this chapter indicates that more discriminative information can be extracted from annotators when they manifest their doubts by providing candidate sets instead of single labels. To establish a baseline in the standard full labelling framework to compare with, the key works by Dawid-Skene [24] and Raykar et al. [13] are used. Empirical results show that, compared to these classical approaches, our methods need fewer and/or lower-quality annotators to obtain classifiers with the same quality. This indicates that candidate labelling could make it cheaper to obtain supervised data through crowdsourcing. According to the experimental results, on average, joint aggregation and learning (JL-C) outperforms the sequential approach (SL-C). Nevertheless, the differences are limited: they change from one domain to another and cannot be attributed to any experimental factor. This work was published in the journal *IEEE Intelligent Systems* [69].

5

General Conclusions and Future Work

5.1 General conclusions

In this thesis, we have proposed three improvements within the learning from crowds framework. The global objective of this thesis was to design strategies that allow saving costs in the learning from crowds framework. In Section 1.4, we briefly introduced the three specific objectives of this thesis. Then, each of them was explained in detail in Chapters 2, 3 and 4, respectively.

The first objective is related to the use of the explanatory variable for improving efficiency in the label aggregation process. In Chapter 2 we proposed *domain-aware voting* (DAV). DAV is a label aggregation procedure that introduces the information from the descriptive feature into a voting scheme as a weighted extra vote, called *domain vote*. This extra vote can be obtained through different means (e.g., the predictions of a classifier) and can help to break ties or complete the information when there are few or no labels in a part of the training set. The domain vote has increased relevance in cases where there is higher uncertainty in the crowdsourced labelling, so DAV acquires a self-regulatory behaviour.

The gathered empirical evidence shows that DAV obtains better performance than MV in terms of aggregation accuracy in the majority of cases. Moreover, there is a notorious advantage of DAV over MV in scenarios where the label distribution is not uniform. The difference between the accuracy values is also high when there is a general lack of labels or with unreliable annotators. Note that the domain vote does not entail an additional monetary cost and that it can serve to obtain similar results to MV with fewer annotators. These facts indicate that DAV uses the available labels more efficiently and thus is useful for saving costs within the learning from crowds framework. DAV was published in the journal *Knowledge and Information Systems* [55].

The second contribution is a flexible labelling scheme that allows for extracting more information from the annotators than traditional full labelling. In this new labelling framework annotators are allowed to express their doubts about the labels they provide, in contrast to the traditional full labelling framework where they are forced to provide a single label. This was stated as our second objective. To fulfil it, we proposed in Chapter 3 the concept of candidate labelling for crowd learning, that is, letting annotators provide a set of labels for each instance. In that way, labellers that are hesitating between various labels can express their doubts. We proposed the extension of MV to the candidate labelling framework, which we call candidate voting (CV). We performed a comparison between the two labelling frameworks using the simple voting methods MV and CV. In those experiments, we observed that candidate labelling can be especially useful in cases where there are few available annotators, difficult instances, a high number of classes or doubtful annotators. Its use can help save costs in the labelling process, as fewer annotators are required for obtaining similar results than in the full labelling framework.

In the same chapter, we proposed a simple annotator model for the candidate labelling context that leads to a label aggregation technique based on the EM strategy. The enhanced performance of the EM-based method regarding the simple CV and approval voting techniques suggests that modelling annotators is possible in this context too. It obtains an advantage when the number of annotators is low and their levels of reliability are varied. These results imply that competitive results can be reached with fewer crowdsourced labels than using full labelling, involving a lower cost. One part of this work was made available in a public repository [66], and another one was presented at the conference *Conference of the Spanish Association of Artificial Intelligence*, CAEPIA 2018 [67].

Candidate labelling opened a new fruitful research line. Once the potential advantages of candidate labelling for aggregation were observed, our third objective was to tackle the problem of learning a classifier from data labelled with multiple candidate sets. In Chapter 4, we proposed a sequential learning method, SL-C, which aggregates the labels before learning the classifier. We also proposed a joint learning method, JL-C, which aggregates the labels and learns the classifier simultaneously. Firstly, we presented a more sophisticated annotator reliability model which accounts for the confusion that annotators can have between any two class labels. The two developed learning methods, SL-C and JL-C, are inspired by two popular learning techniques from the full labelling framework: Dawid-Skene [24], presented in Section 4.2.2, and Raykar et al. [13], presented in Section 4.3, respectively.

The gathered empirical evidence suggests that the methods that use candidate labelling are more cost-effective than the ones using full labelling. Our proposals seem to need fewer and/or lower reliability annotators than the baselines to obtain similar performance. This fulfils the principal objective, which is to reduce costs in the labelling process. There is no consistent evidence that the sequential learning strategy outperforms the joint learning strategy or vice versa. This work was published in the *IEEE Intelligent Systems* journal [69].

5.2 Future work

The contributions of this thesis open the door for further research on making crowd-learning more efficient. Steps forward can be made mainly on different uses of the presented DAV method, creation of databases and techniques based on candidate labelling and development of new labelling schemes. Below we discuss several potential future paths.

- **Study on the weighing parameter of DAV**
  As discussed in Section 2.4, it would be interesting to study how to select a value for the weighing parameter ω_0 depending on the crowdsourced data. An automated method for selecting an adequate ω_0 parameter that gives more (less) importance to the domain vote when there are lower (higher) chances that the crowdsourced labels alone offer a correct estimate of the ground truth could be developed.

- **New domain voters for DAV with different methods.**
  DAV, apart from being easy to implement, can be combined with different types of domain votes, being able to adapt to datasets of different characteristics. It is used with different classifiers in Chapter 2 as domain voters: Random forest, logistic regression and k-nearest neighbours. Employing different domain voters other than a classifier depending on the domain could lead to better results. For example, prior probabilities or density estimation based on previously observed data could be considered as domain votes. Further study in that direction could help decide which domain voter to use in each case.

- **DAV as the base of more complex methods.**
  MV is implicitly or explicitly the base of many crowd-learning methods. DAV could be introduced into more complex aggregation or learning methods where a type of voting is performed, for example as a part of the E-step in EM-based algorithms or in weighted voting techniques. Also, DAV can be combined with the methods proposed for candidate labelling by adapting it to that framework.

- **Construction of datasets based on candidate labelling.**
  As candidate labelling is a novel proposal for crowdsourced data labelling, there are still no datasets labelled following that scheme. This is why we resorted to synthetic data in Chapters 3 and 4. If labels were gathered for several datasets through candidate labelling, practitioners could benefit from substantial savings in the cost of annotating large databases. Also, the generation of candidate labelling-based datasets would allow us to validate our techniques on real data. Furthermore, it could boost the

research in this line, attracting more researchers and leading to the proposal of new label aggregation and classifier learning methods.

- **Refining the set of available labels in an active learning framework.**
  The candidate labelling scheme can be combined with active learning techniques in the same way as it is done with full labelling. A way of selecting the instances that need more labels could be studied. Also, an interesting direction would be to refine the set of possible labels shown to each annotator as more labels are gathered. That set could be reduced based on the labels already provided by previous annotators.

- **Selection of an adequate learning scheme**
  It remains an open question in which experimental scenarios should be preferred for the use of sequential or joint learning approaches for the candidate labelling framework, as our experimental results from Section 4.5.3 do not show any general trend. A broader specific study to answer this question would be useful for practitioners.

- **Exploration of new types of flexible labelling**
  The evidence gathered in Chapters 3 and 4 suggests that allowing annotators to provide fine-grained information regarding their knowledge can be beneficial. It enables to build learning methods that obtain similar results by counting on more reduced resources. Thus, it is reasonable to think that it would be interesting to develop more flexible labelling schemes than candidate labelling. For example, in candidate labelling annotators do not distinguish between the labels provided in each candidate set. However, an annotator could be more confident about some labels than about others. In that sense, giving the option of providing a ranking of the selected labels would offer more information about the knowledge of the annotators. Furthermore, an annotator could weigh their confidence in each class label, for example, by providing a probability distribution over the class labels.
  Candidate labelling stands out as it is generally less challenging than full labelling itself, as the annotators do not have to deliberate to provide a single label if they are not sure about it. In contrast, as the labelling scheme gets more and more refined, it could also become more demanding for the annotators in terms of time and effort, which could make the labelling process more costly. Even so, this could be a promising research line, as more information can be extracted despite compromising the simplicity of the labelling process. The trade-off between the gain in performance and the loss in time or money should be studied. In that way, one can explore new forms of collecting labels where annotators can express their opinions in different ways.

Another characteristic of flexible labelling schemes is that they generalise other labelling schemes that are more constraining. For example, with candidate labelling, annotators can still provide a single label for an instance as in full labelling. In a framework where labellers use a probability distribution over the class labels, they could assign the same probability to a subset of labels, which is equivalent to candidate labelling. They could also assign probability 1 to just one class label, which would be full labelling. Thus, the use of flexible schemes allows annotators to choose which level of information they want to provide.

5.3 Main achievements

Journal papers

- Beñaran-Muñoz, I., Hernández-González, J. & Pérez, A. "On the use of the descriptive variable for enhancing the aggregation of crowdsourced labels" Knowledge and Information Systems, vol. 65, pp. 241-260 (2023). DOI: https://doi.org/10.1007/s10115-022-01743-z
- Beñaran-Muñoz, I., Hernández-González, J., & Pérez, A. "Machine learning from crowds using candidate set-based labelling" IEEE Intelligent Systems, vol. 37, no. 6, pp. 57-68 (2022). DOI: https://doi.org/10.1109/MIS.2022.3205053

Conferences

- Beñaran-Muñoz, I., Hernández-González, J., & Pérez, A. "Crowd Learning with Candidate Labeling: an EM-based Solution". In proceedings of 18th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2018), Granada, Spain. (Oral presentation)
- Beñaran-Muñoz, I., Hernández-González, J., & Pérez, A. "Weak Labeling for Crowd Learning". In the 5th AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2017), Quebec, Canada. (Poster)

Public reports

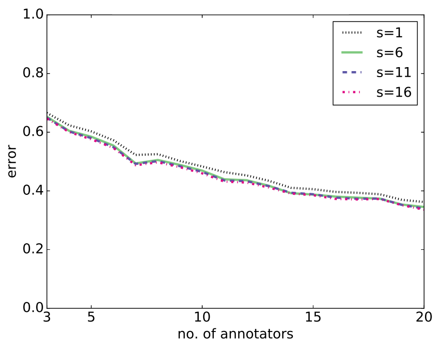
- Beñaran-Muñoz, I., Hernández-González, J., & Pérez, A. "Candidate Labeling for Crowd Learning". Arxiv, 2018. Retrieved from: https://arxiv.org/abs/1804.10023

A

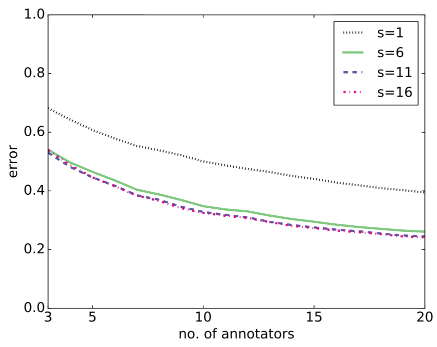
Additional results for comparison between candidate and full labelling with voting techniques

In Figures A.1 to A.4, results of the total error (Eq. 3.4) obtained with additional experimental configurations (combinations of values of r, δ_0 and μ_0) that are not shown in Section 3.3.3 are displayed.

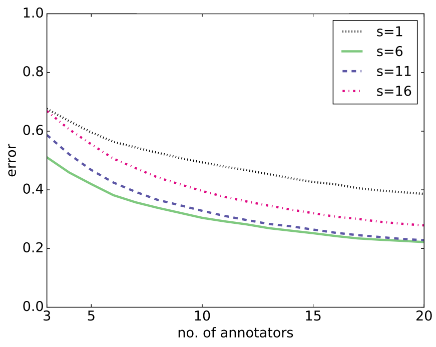
In Figures A.5 to A.7, results of the decomposition of the error obtained with additional experimental configurations (combinations of values of r, δ_0 and l) that are not shown in Section 3.3.3 are displayed.



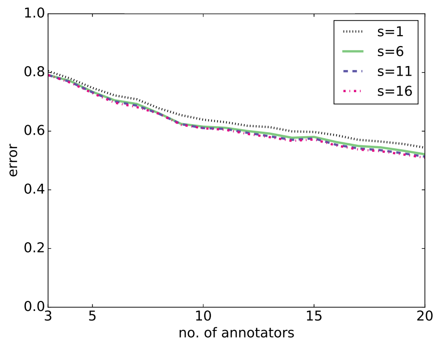
(a) $r = 32,\ \delta = 0.2,\ \mu_0 = 0.125$



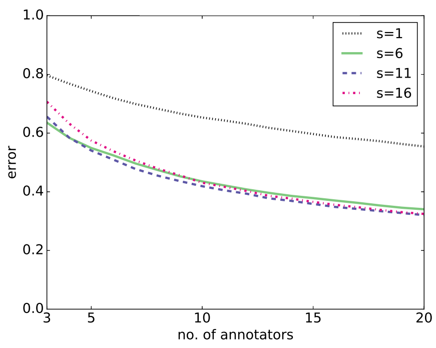
(b) $r = 32,\ \delta = 0.2,\ \mu_0 = 4$



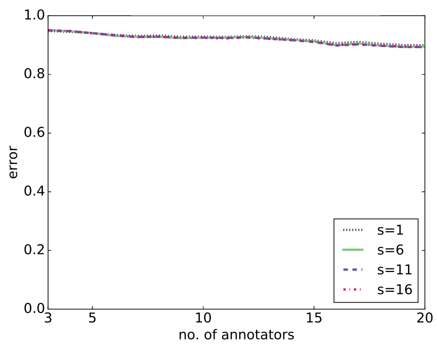
(c) $r = 32,\ \delta = 0.2,\ \mu_0 = 32$



(d) $r = 32,\ \delta = 0.5,\ \mu_0 = 0.125$

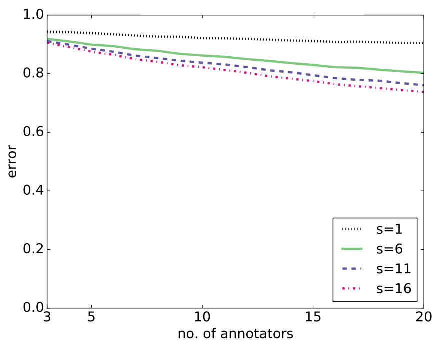


(e) $r = 32,\ \delta = 0.5,\ \mu_0 = 32$

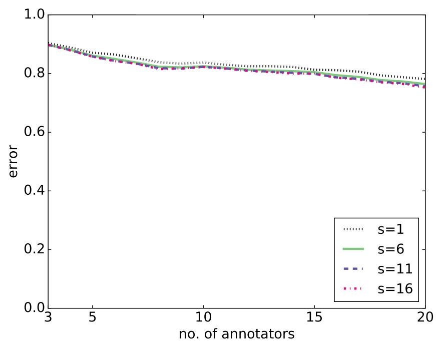


(f) $r = 32,\ \delta = 10,\ \mu_0 = 0.125$

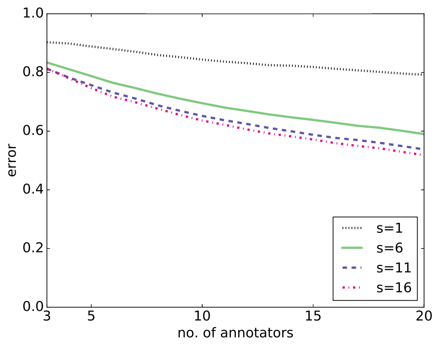
Fig. A.1: Graphical description of the error (Eq. 3.4) obtained by annotation simulated with different values of r, δ and μ_0. Error curves for different values of s are shown in each figure.



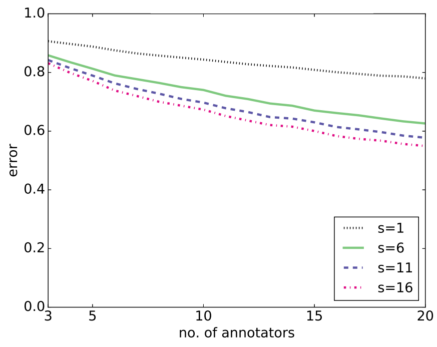
(a) $r = 32,\ \delta = 10,\ \mu_0 = 32$



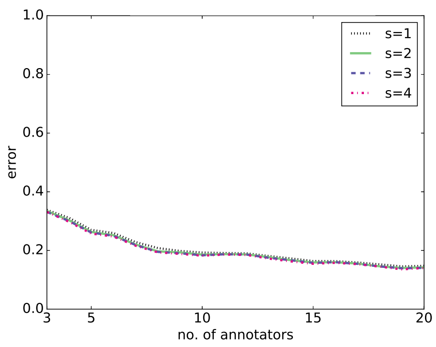
(b) $r = 32,\ \delta = 2,\ \mu_0 = 0.125$



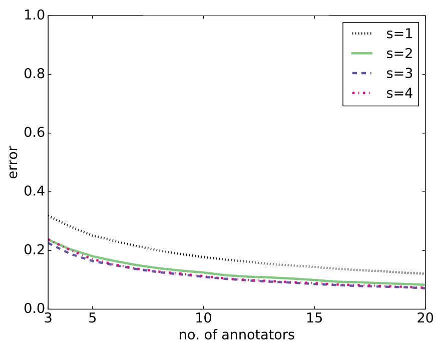
(c) $r = 32,\ \delta = 2,\ \mu_0 = 32$



(d) $r = 32,\ \delta = 2,\ \mu_0 = 4$

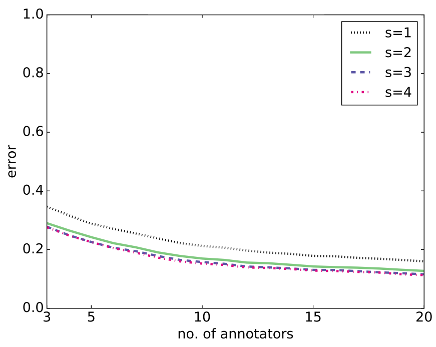


(e) $r = 8,\ \delta = 0.2,\ \mu_0 = 0.125$

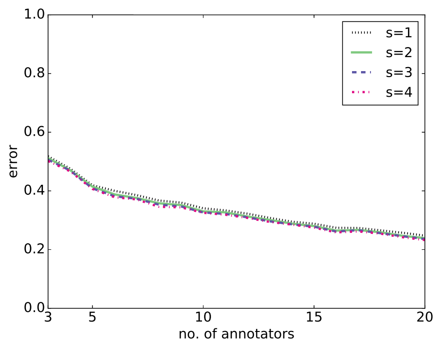


(f) $r = 8,\ \delta = 0.2,\ \mu_0 = 32$

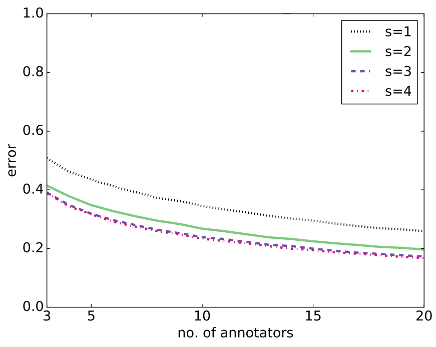
Fig. A.2: Graphical description of the error (Eq. 3.4) obtained by annotation simulated with different values of r, δ and μ_0. Error curves for different values of s are shown in each figure.



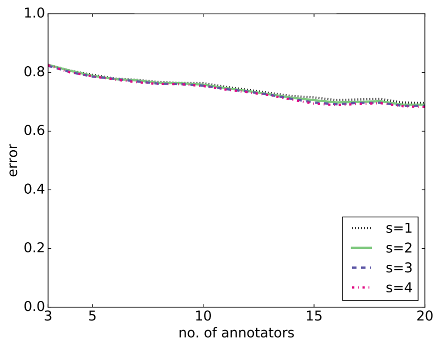
(a) $r = 8,\ \delta = 0.2,\ \mu_0 = 4$



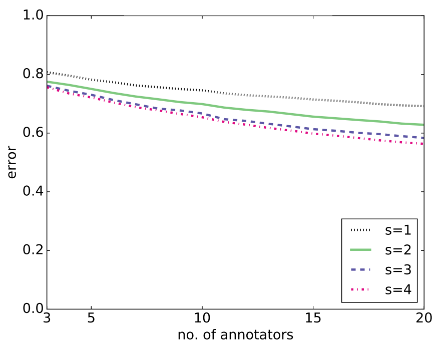
(b) $r = 8,\ \delta = 0.5,\ \mu_0 = 0.125$



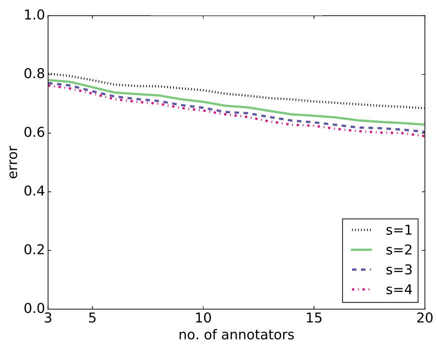
(c) $r = 8,\ \delta = 0.5,\ \mu_0 = 32$



(d) $r = 8,\ \delta = 10,\ \mu_0 = 0.125$

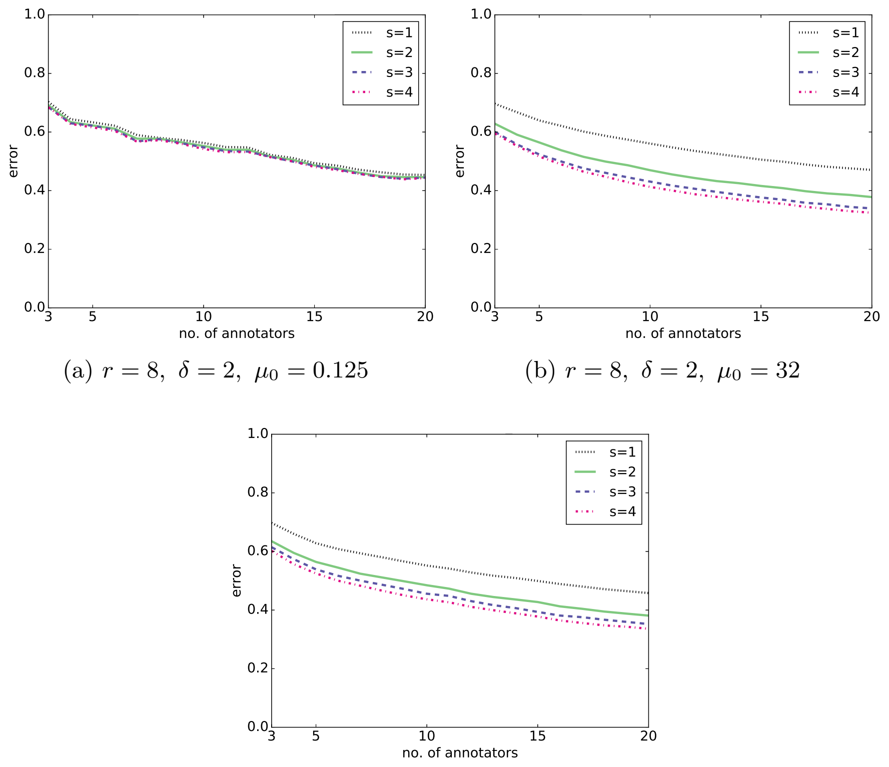


(e) $r = 8,\ \delta = 10,\ \mu_0 = 32$



(f) $r = 8,\ \delta = 10,\ \mu_0 = 4$

Fig. A.3: Graphical description of the error (Eq. 3.4) obtained by annotation simulated with different values of r, δ and μ_0. Error curves for different values of s are shown in each figure.

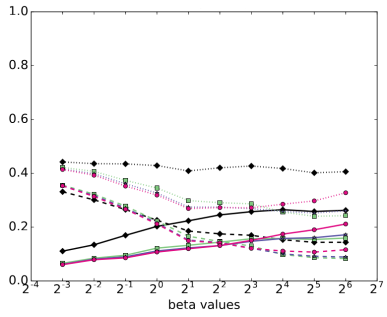


(a) $r = 8,\ \delta = 2,\ \mu_0 = 0.125$

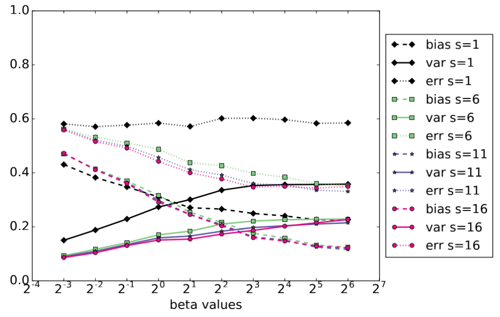
(b) $r = 8,\ \delta = 2,\ \mu_0 = 32$

(c) $r = 8,\ \delta = 2,\ \mu_0 = 4$

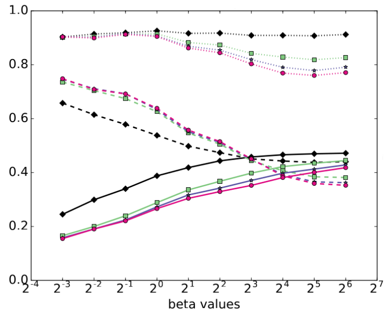
Fig. A.4: Graphical description of the error (Eq. 3.4) obtained by annotation simulated with different values of r, δ and μ_0. Error curves for different values of s are shown in each figure.



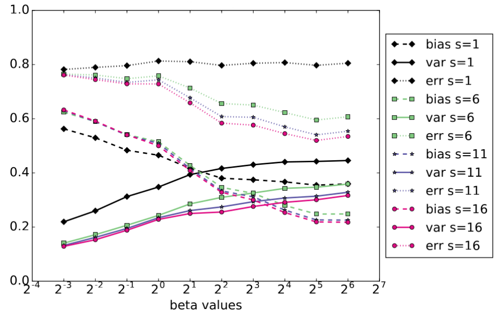
(a) $r = 32,\ \delta = 0.2,\ l = 16$



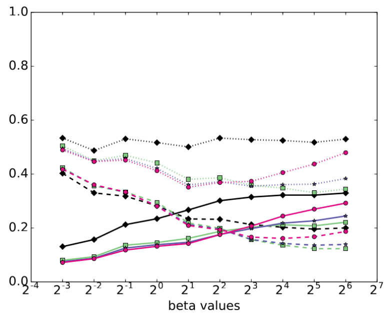
(b) $r = 32,\ \delta = 0.5,\ l = 16$



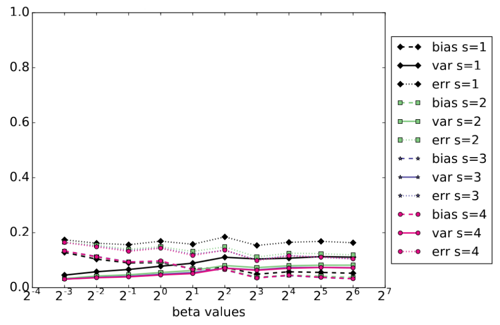
(c) $r = 32,\ \delta = 10,\ l = 16$



(d) $r = 32,\ \delta = 2,\ l = 16$

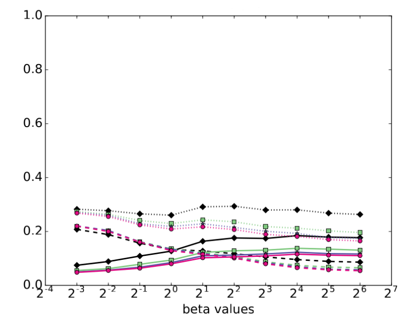


(e) $r = 32,\ \delta = 0.2,\ l = 8$

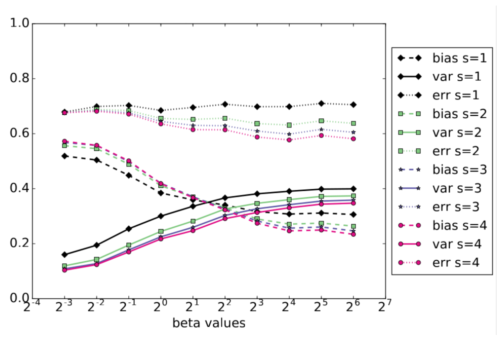


(f) $r = 8,\ \delta = 0.2,\ l = 16$

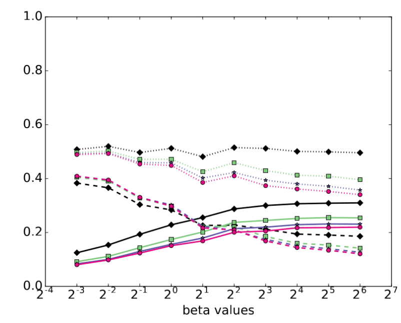
Fig. A.5: Graphical description of the decomposition of the error obtained by annotation simulated with different values of the parameters. Curves representing the squared bias, the variance and the total error are displayed in each figure.



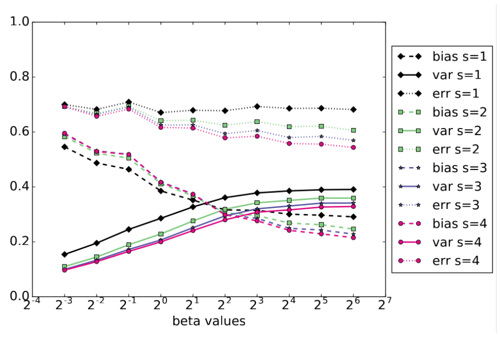
(a) $r = 8,\ \delta = 0.5,\ l = 16$



(b) $r = 8,\ \delta = 10,\ l = 16$

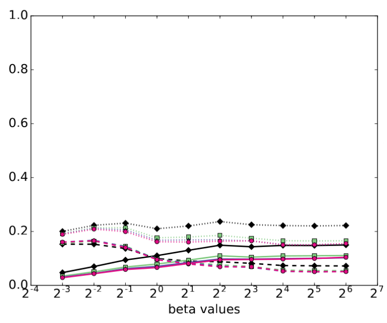


(c) $r = 8,\ \delta = 2,\ l = 16$

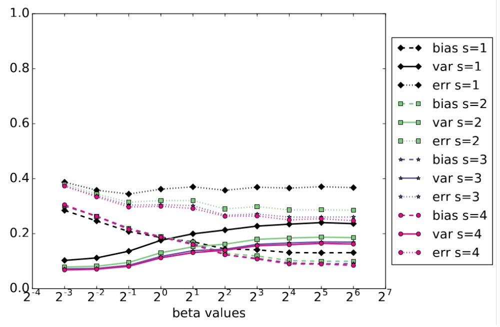


(d) $r = 8,\ \delta = 10,\ l = 20$

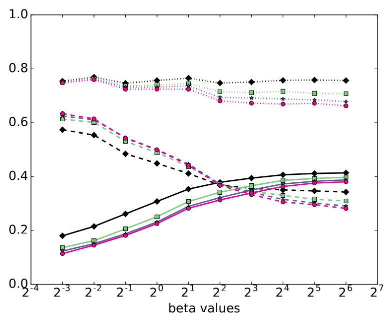
Fig. A.6: Graphical description of the decomposition of the error obtained by annotation simulated with different values of the parameters. Curves representing the squared bias, the variance and the total error are displayed in each figure.



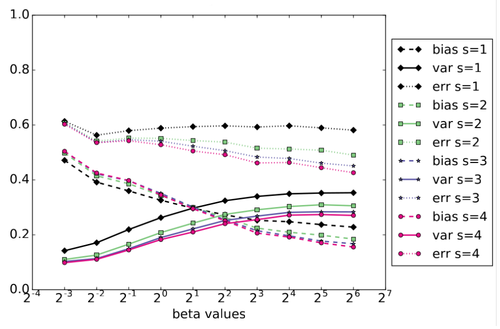
(a) $r = 8,\ \delta = 0.2,\ l = 8$



(b) $r = 8,\ \delta = 0.5,\ l = 8$



(c) $r = 8,\ \delta = 10,\ l = 8$



(d) $r = 8,\ \delta = 2,\ l = 8$

Fig. A.7: Graphical description of the decomposition of the error obtained by annotation simulated with different values of the parameters. Curves representing the squared bias, the variance and the total error are displayed in each figure.

B

Additional results for comparison between candidate and full labelling with EM-based techniques

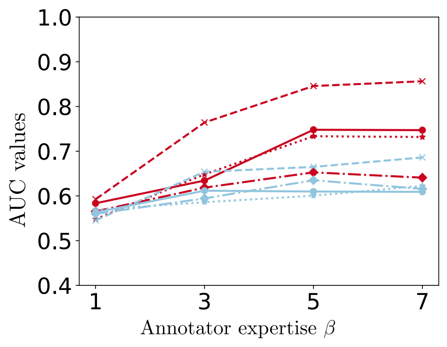
Figures with additional datasets

In Figures B.1, B.2 and B.3, results obtained with other datasets that are not shown in the manuscript are displayed. Results with 3 supervised datasets from UCI repository (http://archive.ics.uci.edu/ml) are shown: *Arrhythmia* $(452, 13)$, *Pendigits* $(10992, 10)$ and *Satimage* $(6435, 6)$, with numbers meaning no. instances n, and no. classes r.

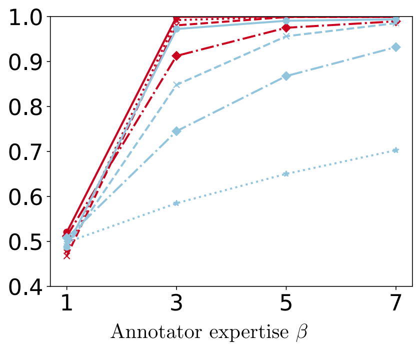
We simulate different numbers of annotators $m \in \{3, 5, 7, 9\}$, and different degrees of expertise for them $\beta \in \{1, 3, 5, 7\}$. For candidate labelling generation, the proportion of sampled labels takes values $prop \in \{0.1, 0.3, 0.5, 0.7\}$. We have used two classifiers from very different families from *sklearn 0.22.1* with default parameters: 5-Nearest Neighbour (5NN) and Random Forest (RF).

The models are evaluated using the area under the ROC curve (AUC). It is estimated using stratified 5-fold cross-validation, where the test sets are fully supervised.

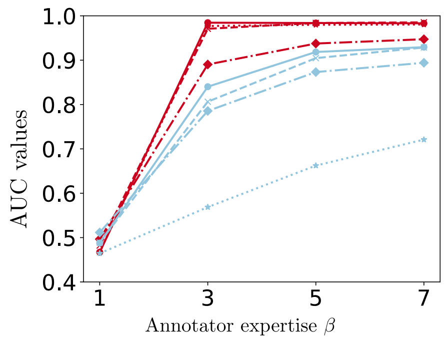
Figure B.1 shows the impact of the expertise (β parameter) of the annotators on the performance of the methods, Figure B.2 shows the impact of the number of annotators (m) on the performance of the methods, and Figure B.3 shows the effect of the maximum candidate set size ($prop$) in the performance.



(a) Dataset *arrhythmia*



(b) Dataset *pendigits*



(c) Dataset *satimage*

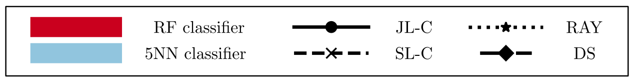
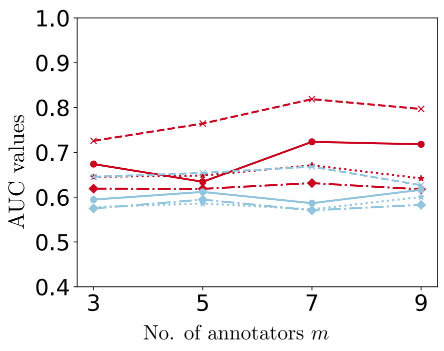
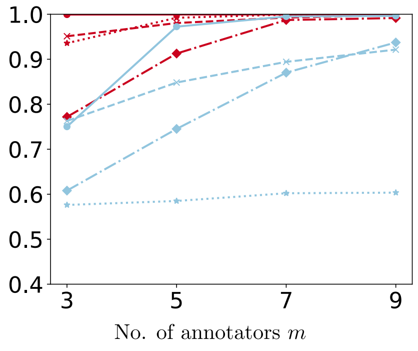


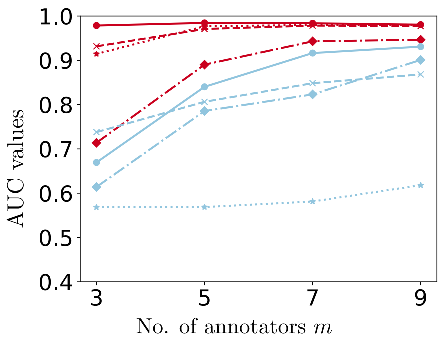
Fig. B.1: Experimental results throughout different values of the parameter β (annotator expertise), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in orange and red colours, respectively. A different line style and marker are used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $m = 5$ and $prop = 0.5$.



(a) Dataset *arrhythmia*



(b) Dataset *pendigits*



(c) Dataset *satimage*

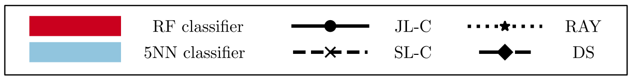
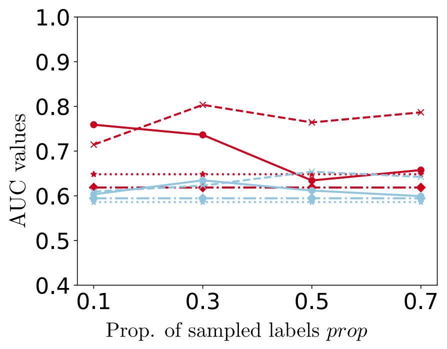
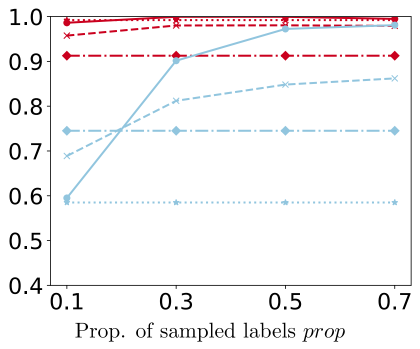


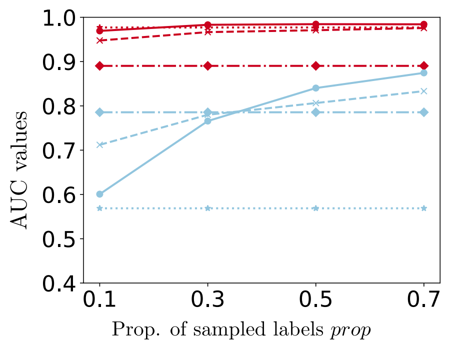
Fig. B.2: Experimental results throughout different values of the parameter m (number of annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in orange and red colours, respectively. A different line style and marker are used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 3$ and $prop = 0.5$.



(a) Dataset *arrhythmia*



(b) Dataset *pendigits*



(c) Dataset *satimage*

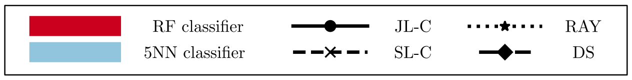


Fig. B.3: Experimental results throughout different values of the parameter *prop* (flexibility of the annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in orange and red colours, respectively. A different line style and marker are used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 3$ and $m = 5$.

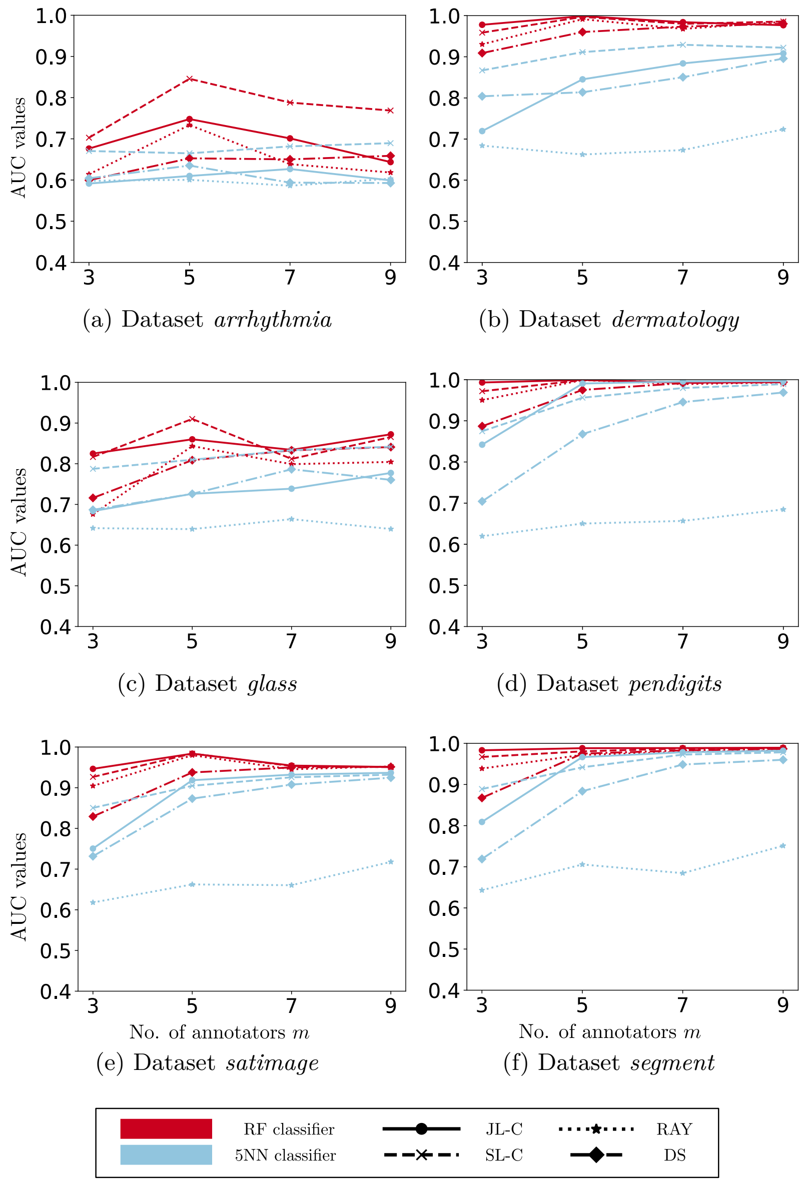
Figures with additional configurations

In Figures B.1, B.2 and B.3, results obtained with the fixed parameter value $\beta = 5$ are shown. We use 9 fully labelled datasets from UCI repository (http://archive.ics.uci.edu/ml): *Arrhythmia* $(452, 13)$, *Dermatology* $(366, 6)$, *Glass* $(214, 6)$, *Pendigits* $(10992, 10)$, *Satimage* $(6435, 6)$, *Segment* $(2310, 7)$, *Svmguide4* $(612, 6)$, *Vehicle* $(846, 4)$, and *Vowel* $(990, 11)$, and 3 partially labelled datasets [70] []: *Birdac* $(3718, 13)$, *Lost* $(1122, 14)$ and *MSRCv2* $(1758, 23)$, with numbers meaning number of instances n, and number of classes r.

We simulate different numbers of annotators $m \in \{3, 5, 7, 9\}$, and different degrees of expertise for them $\beta \in \{1, 3, 5, 7\}$. For candidate labelling generation, the proportion of sampled labels takes values $prop \in \{0.1, 0.3, 0.5, 0.7\}$. We have used two classifiers from very different families from *sklearn 0.22.1* with default parameters: 5-Nearest Neighbour (5NN) and Random Forest (RF).

The models are evaluated using the area under the ROC curve (AUC). It is estimated using stratified 5-fold cross-validation, where the test sets are fully supervised.

Figures B.4 and B.5 show the impact of the number of annotators (m) on the performance of the methods, and Figures B.6 and B.7 show the effect of the maximum candidate set size ($prop$) in the performance.



(a) Dataset *arrhythmia* (b) Dataset *dermatology*

(c) Dataset *glass* (d) Dataset *pendigits*

(e) Dataset *satimage* (f) Dataset *segment*

Fig. B.4: Experimental results throughout different values of the parameter m (number of annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 5$ and $prop = 0.5$.

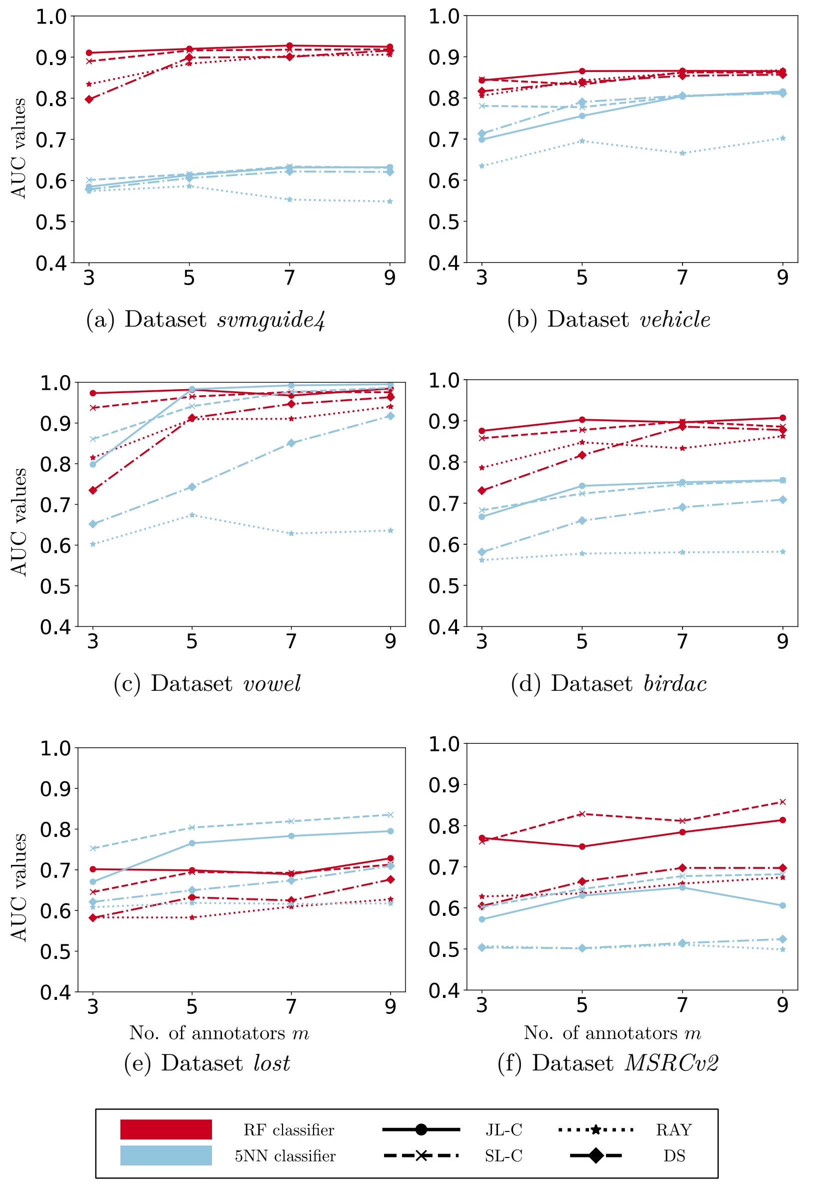
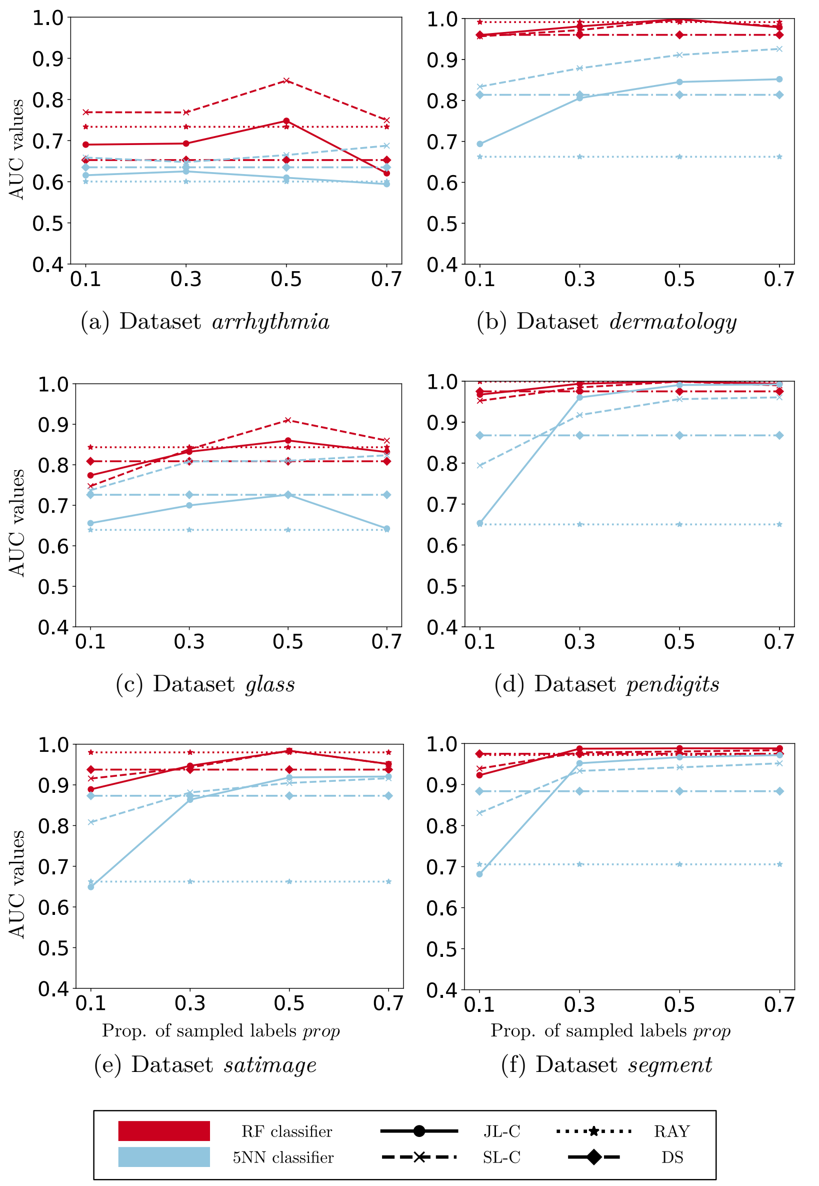


Fig. B.5: Experimental results throughout different values of the parameter m (number of annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 5$ and $prop = 0.5$.

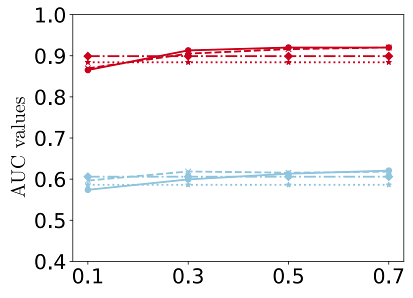


(a) Dataset *arrhythmia* (b) Dataset *dermatology*

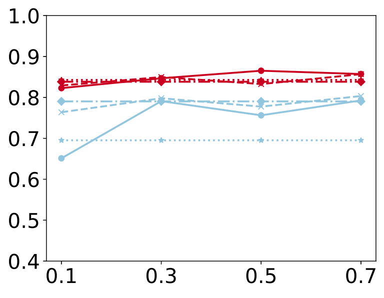
(c) Dataset *glass* (d) Dataset *pendigits*

(e) Dataset *satimage* (f) Dataset *segment*

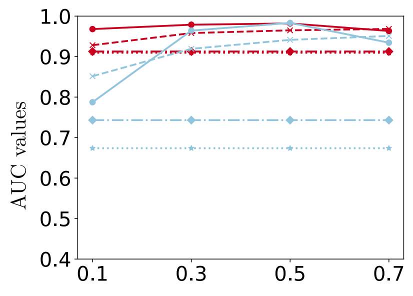
Fig. B.6: Experimental results throughout different values of the parameter *prop* (flexibility of the annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 5$ and $m = 5$.



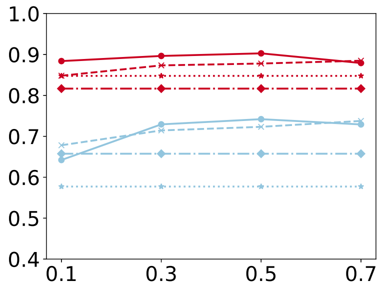
(a) Dataset *svmguide4*



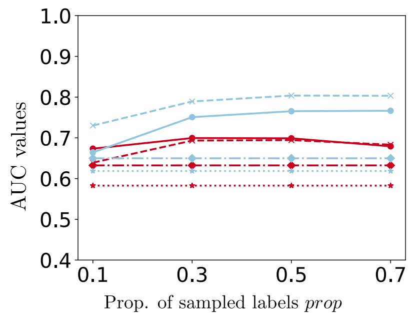
(b) Dataset *vehicle*



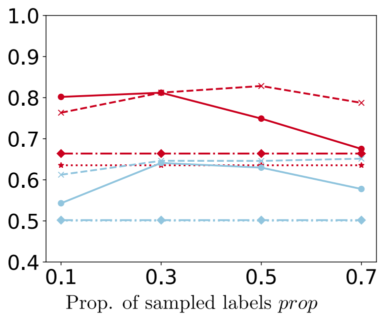
(c) Dataset *vowel*



(d) Dataset *birdac*



(e) Dataset *lost*



(f) Dataset *MSRCv2*

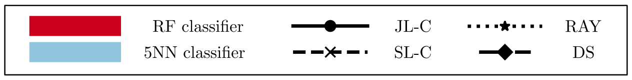


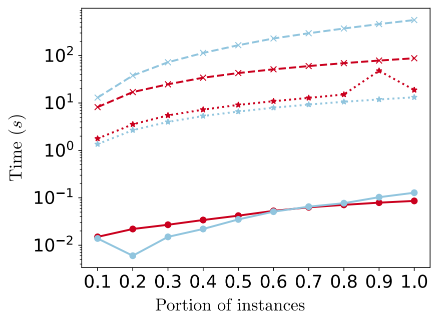
Fig. B.7: Experimental results throughout different values of the parameter *prop* (flexibility of the annotators), in terms of AUC metric, within different datasets (subplots). Results with classifiers RF and 5NN are displayed in dark blue and light blue colour, respectively. A different line style and marker is used for each method (SL-C, JL-C, RAY, DS). The rest of the generative parameters are fixed to $\beta = 5$ and $m = 5$.

C

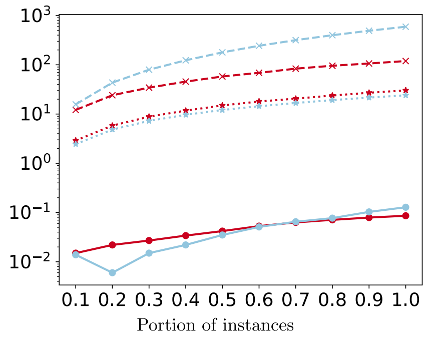
Scalability test with methods SL-C and JL-C

We have performed a scalability test by using subsets of an increasing size of the *pendigits* dataset (the largest one among the considered datasets) and applying our methods SL-C and JL-C on them. Each subset has a portion of the original instances, ranging from 0.1 (10% of the instances are preserved) to 1 (the entire dataset is used). Stratified sampling is performed to keep the class proportions. The time taken for a run of the EM method is calculated, using RF and 5NN as base classifiers. The learning time of the classifiers themselves is also included for comparison. The number of annotators is varied between 3 and 6 to see its effect on the execution time. Also, apart from considering all 6 class labels from the dataset, the execution time considering only half of the class labels (3) is also computed to understand the impact of the size of the class variable.

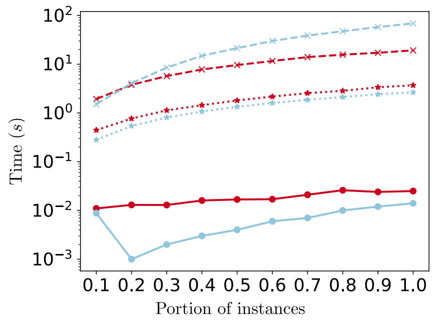
The results of this scalability test can be observed in Figure C.1. The numbers of instances, classes and annotators affect the running times of both of our methods, and so does the choice of the classifier. The running time of SL-C is always lower than that of JL, which seems to grow exponentially when employing RF and linearly in the case of 5NN. When doubling the number of annotators, the running time of our methods has only a small increase. However, when the number of classes is reduced to half, the difference is notorious, reducing the running time to nearly 8 or 10 times for JL-C with RF, and about 4 times for SL-C with 5NN.



(a) 6 classes, 3 annotators



(b) 6 classes, 6 annotators



(c) 3 classes, 3 annotators

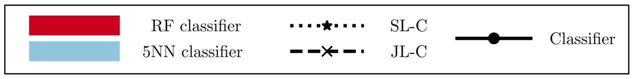


Fig. C.1: Scalability test with our two methods, SL-C and JL-C, and the classifiers RF and 5NN, throughout different portions of the complete dataset. Running time is measured in seconds and shown in a logarithmic scale on the Y axis. Results with classifiers RF and 5NN are displayed in red and light blue colour, respectively. Different line styles and markers are used for each method. Each subfigure shows the performance with varying numbers of classes and annotators.

References

[1] Steve Hanneke. The optimal sample complexity of pac learning. *The Journal of Machine Learning Research*, 17(1):1319–1333, 2016.

[2] Richard Y Wang, Veda C Storey, and Christopher P Firth. A framework for analysis of data quality research. *IEEE transactions on knowledge and data engineering*, 7(4):623–640, 1995.

[3] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2, 2017.

[4] Jeff Howe et al. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

[5] Menatalla Abououf, Hadi Otrok, Rabeb Mizouni, Shakti Singh, and Ernesto Damiani. How artificial intelligence and mobile crowd sourcing are inextricably intertwined. *IEEE Network*, 35(3):252–258, 2020.

[6] Miguel Angel Luengo-Oroz, Asier Arranz, and John Frean. Crowdsourcing malaria parasite quantification: an online game for analyzing images of infected thick blood smears. *J. Med. Internet Res.*, 14(6):e167, 2012.

[7] Jonathan Silvertown. A new dawn for citizen science. *Trends in ecology & evolution*, 24(9):467–471, 2009.

[8] Jesus Cerquides, Mehmet Oğuz Mülâyim, Jerónimo Hernández-González, Amudha Ravi Shankar, and Jose Luis Fernandez-Marquez. A conceptual probabilistic framework for annotation aggregation of citizen science data. *Mathematics*, 9(8):875, 2021.

[9] Hui Gao, Chi Harold Liu, Wendong Wang, Jianxin Zhao, Zheng Song, Xin Su, Jon Crowcroft, and Kin K Leung. A survey of incentive mechanisms for participatory sensing. *IEEE Communications Surveys & Tutorials*, 17(2):918–943, 2015.

[10] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. Crowdsourced Data Management: A Survey. *IEEE TKDE*, 28(9):2296–2319, 2016.

[11] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast - but is it good? evaluating non-expert annotations for

natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

[12] Victor S. Sheng, Foster J. Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the Special Interest Group on Knowledge Discovery and Data Mining*, pages 614–622, 2008.

[13] V. C. Raykar, S. Yu, L. H. Zhao, G. H Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

[14] Jerónimo Hernández-González, Iñaki Inza, and Jose A. Lozano. Weak supervision and other non-standard classification problems: A taxonomy. *Pattern Recognition Letters*, 69:49–55, 2016. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2015.10.008.

[15] Timothée Cour, Benjamin Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12:1501–1536, 2011.

[16] Victor S Sheng, Jing Zhang, Bin Gu, and Xindong Wu. Majority voting and pairing with multiple noisy labeling. *IEEE Transactions on Knowledge and Data Engineering*, 31(7):1355–1368, 2017.

[17] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc. Series B (Methodological)*, 39(1):1–38, 1977.

[18] Geoffrey J McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[19] Victor S Sheng and Jing Zhang. Machine learning with crowdsourcing: A brief summary of the past research and future directions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9837–9843, 2019.

[20] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. *Communications of the Association for Computing Machinery*, 58(8):85–94, 2015.

[21] Jonathan Corney, Andrew Lynn, Carmen Torres, Paola Di Maio, William Regli, Graeme Forbes, and Lynne Tobin. Towards crowdsourcing translation tasks in library cataloguing, a pilot study. In *IEEE International Conference on Digital Ecosystems and Technologies*, pages 572–577. IEEE, 2010.

[22] Kerri Wazny. Applications of crowdsourcing in health: an overview. *Journal of Global Health*, 8(1), 2018.

[23] Enrique G. Rodrigo, Juan A Aledo, and José A Gámez. Machine learning from crowds: A systematic review of its applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(2):e1288, 2019.

[24] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society Series C (Applied Statistics)*, 28(1):20–28, 1979.

[25] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Neural Information Processing Systems*, pages 2035–2043, 2009.

[26] David R Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Neural Information Processing Systems*, pages 1953–1961, 2011.

[27] Y. et al. Yan. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of AISTATS*, pages 932–939, 2010.

[28] Jing Zhang, Victor S Sheng, and Jian Wu. Crowdsourced label aggregation using bilayer collaborative clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 30(10):3172–3185, 2019.

[29] Victor S Sheng. Simple multiple noisy label utilization strategies. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 635–644. IEEE, 2011.

[30] Filipe Rodrigues and Francisco Pereira. Deep learning from crowds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[31] Ryutaro Tanno, Ardavan Saeedi, Swami Sankaranarayanan, Daniel C Alexander, and Nathan Silberman. Learning from noisy labels by regularized estimation of annotator confusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11244–11253, 2019.

[32] Jing Zhang, Xindong Wu, and Victor S Sheng. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576, 2016.

[33] Behrooz Parhami. Voting algorithms. *IEEE transactions on reliability*, 43(4):617–629, 1994.

[34] Ludmila I Kuncheva, Christopher J Whitaker, Catherine A Shipp, and Robert PW Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis & Applications*, 6(1):22–31, 2003.

[35] Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G Dy. Active learning from crowds. In *International Conference on Machine Learning*, volume 11, pages 1161–1168, 2011.

[36] Joel Ross, Lilly Irani, M Six Silberman, Andrew Zaldivar, and Bill Tomlinson. Who are the crowdworkers? shifting demographics in mechanical turk. In *CHI'10 extended abstracts on Human factors in computing systems*, pages 2863–2872. ACM, 2010.

[37] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 239–254, 2014.

[38] Minghong Fang, Minghao Sun, Qi Li, Neil Zhenqiang Gong, Jin Tian, and Jia Liu. Data poisoning attacks and defenses to crowdsourcing systems. In *Proceedings of the Web Conference 2021*, pages 969–980, 2021.

[39] Djellel Eddine Difallah, Gianluca Demartini, and Philippe Cudré-Mauroux. Mechanical cheat: Spamming schemes and adversarial techniques on crowdsourcing platforms. In *CrowdSearch*, 2012.

[40] Ludmila I Kuncheva and Juan J Rodríguez. A weighted voting framework for classifiers ensembles. *Knowledge and information systems*, 38(2):259–275, 2014.

[41] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Proceedings of Advances in Neural Information Processing Systems 23 (NIPS)*, pages 2424–2432, 2010.

[42] Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. *The Journal of Machine Learning Research*, 17(1):3537–3580, 2016.

[43] Vaibhav B Sinha, Sukrut Rao, and Vineeth N Balasubramanian. Fast dawid-skene: A fast vote aggregation scheme for sentiment classification. *arXiv preprint arXiv:1803.02781*, 2018.

[44] Yi Yang, Quan Bai, and Qing Liu. Modeling random guessing and task difficulty for truth inference in crowdsourcing. In *AAMAS*, volume 19, pages 2288–2290, 2019.

[45] Y. et al. Yan. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of AISTATS*, pages 932–939, 2010.

[46] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951.

[47] A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

[48] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[49] Juan D Rodríguez, Aritz Pérez, and Jose A Lozano. A general framework for the statistical analysis of the sources of variance for classification error estimators. *Pattern Recognition*, 46(3):855–864, 2013.

[50] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[51] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[52] A Urkullu, A Pérez, and B Calvo. On the evaluation and selection of classifier learning algorithms with crowdsourced data. *Applied Soft Computing*, 80:832–844, 2019.

[53] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks and Learning Systems*, 20(3):542–542, 2009.

[54] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[55] Iker Beñaran-Muñoz, Jerónimo Hernández-González, and Aritz Pérez. On the use of the descriptive variable for enhancing the aggregation of crowdsourced labels. *Knowledge and Information Systems*, 65:241–260, 2022. doi: 10.1007/s10115-022-01743-z.

[56] Marina Torre, Shinnosuke Nakayama, Tyrone J Tolbert, and Maurizio Porfiri. Producing knowledge by admitting ignorance: Enhancing data quality through an “i don’t know” option in citizen science. *PloS one*, 14 (2):e0211907, 2019.

[57] Yao-Xiang Ding and Zhi-Hua Zhou. Crowdsourcing with unsure option. *Mach. Learn.*, 107(4):749–766, 2018.

[58] Jinhong Zhong, Ke Tang, and Zhi-Hua Zhou. Active learning from crowds with unsure option. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1061–1068, 2015.

[59] Padhraic Smyth, Usama M. Fayyad, Michael C. Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. In *Proceedings of Advances in Neural Information Processing Systems 7 (NIPS)*, pages 1085–1092, 1994. URL http://papers.nips.cc/paper/949-inferring-ground-truth-from-subjective-labelling-of-venus-images.

[60] Catherine Grady and Matthew Lease. Crowdsourcing document relevance assessment with mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon’s mechanical turk*, pages 172–179. Association for Computational Linguistics, 2010.

[61] S. Ojha A. Banerjee and D. Gurari. Let’s agree to disagree: A meta-analysis of disagreement among crowdworkers during visual question answering. In *GroupSight at HCOMP*, 2017.

[62] Steven J Brams and Peter C Fishburn. Approval voting. *Am. Polit. Sci. Rev.*, 72(3):831–847, 1978.

[63] J-Cl Falmagne and Michael Regenwetter. A random utility model for approval voting. *J Math Psychol*, 40(2):152–159, 1996.

[64] Ariel D Procaccia and Nisarg Shah. Is approval voting optimal given approval votes? In *Proc. of NIPS 28*, pages 1801–1809, 2015.

[65] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[66] I. Beñaran-Muñoz, J. Hernández-González, and A. Pérez. Weak Labeling for Crowd Learning. *ArXiv e-prints*, 2018.