Master Thesis

Access Path Design for Quality Assurance in Crowdsourcing

Author(s):
Pozzetti, Matteo

Publication Date:
2016

Permanent Link:
https://doi.org/10.3929/ethz-a-010725406

Rights / License:
In Copyright - Non-Commercial Use Permitted
Access Path Design for Quality Assurance in Crowdsourcing

Master Thesis
Matteo Pozzetti
September 07, 2016

Advisors: Prof. Dr. Andreas Krause, Prof. Dr. Donald Kossmann, Besmira Nushi
Department of Computer Science, ETH Zürich
Abstract

In this thesis, we study how the problem of worker group discovery relates to two well-known problems in crowdsourcing: answer aggregation and budget allocation. In contrast to previous studies which assume full independence of worker answers, this work is based on the observation that certain groups of crowd contributors share common behaviors and hence their answers are correlated. Being aware of these clusters of workers can be useful not only to predict more accurately the correct answer from the crowd, but also to wisely distribute the available budget.

We investigate various clustering strategies as solutions to the problem of worker group discovery from historical data. Our studies confirm that the most critical challenge to this problem is the high data sparsity that characterizes crowd work. Therefore, we propose to apply such clustering techniques on reduced representations of the historical data based on the heterogeneity of the tasks. In an experimental setup where tasks belong to different categories, worker groups are constructed based on the workers’ average performance in the distinct task categories. We make use of this distinction to introduce a new model which jointly represents: workers, worker groups, and task categories. Next, we also make use of the information on the category of the task to adjust budget allocation. We evaluated our approach on multiple real-world and synthetic datasets which reveal interesting insights on the applicability and the limitations of our approach.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Overview</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Problem Statement</td>
<td>5</td>
</tr>
<tr>
<td><strong>2 Related work and background</strong></td>
<td>9</td>
</tr>
<tr>
<td>2.1 Related work</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Background</td>
<td>11</td>
</tr>
<tr>
<td>2.2.1 Access Path model</td>
<td>11</td>
</tr>
<tr>
<td>2.2.2 CorEx</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Greedy Budget Allocation</td>
<td>15</td>
</tr>
<tr>
<td><strong>3 Access path discovery</strong></td>
<td>17</td>
</tr>
<tr>
<td>3.1 Impact of data sparsity</td>
<td>17</td>
</tr>
<tr>
<td>3.1.1 CorEx with generated data</td>
<td>17</td>
</tr>
<tr>
<td>3.1.2 CorEx on the personality dataset</td>
<td>21</td>
</tr>
<tr>
<td>3.1.3 Comparison with k-means clustering</td>
<td>21</td>
</tr>
<tr>
<td>3.1.4 Discussion about CorEx</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Sparseness reduction via task grouping</td>
<td>25</td>
</tr>
<tr>
<td><strong>4 Task category impact on aggregation and optimization</strong></td>
<td>29</td>
</tr>
<tr>
<td>4.1 Jointly modeling task groups and access paths</td>
<td>29</td>
</tr>
<tr>
<td>4.2 Task-based budget allocation</td>
<td>31</td>
</tr>
<tr>
<td><strong>5 Experiments</strong></td>
<td>33</td>
</tr>
<tr>
<td>5.1 Datasets overview</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Code and libraries used</td>
<td>35</td>
</tr>
<tr>
<td>5.3 Sparseness reduction used</td>
<td>37</td>
</tr>
<tr>
<td>5.4 Experiments on the group discovery algorithm</td>
<td>40</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.1</td>
<td>Synthetic data</td>
<td>40</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Group discovery performance evaluation strategy</td>
<td>41</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Bluebird dataset</td>
<td>42</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Medical dataset</td>
<td>42</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Topic relevance dataset</td>
<td>43</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Weather sentiment dataset</td>
<td>44</td>
</tr>
<tr>
<td>5.4.7</td>
<td>Sport dataset</td>
<td>45</td>
</tr>
<tr>
<td>5.4.8</td>
<td>Conclusions on the group discovery experiments</td>
<td>46</td>
</tr>
<tr>
<td>5.5</td>
<td>Experimental evaluation of TD-APM</td>
<td>46</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Synthetic data</td>
<td>47</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Medical dataset</td>
<td>49</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Topic relevance dataset</td>
<td>51</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Weather dataset</td>
<td>52</td>
</tr>
<tr>
<td>5.5.5</td>
<td>Sport dataset</td>
<td>52</td>
</tr>
<tr>
<td>5.5.6</td>
<td>Conclusions on the TD-APM</td>
<td>53</td>
</tr>
<tr>
<td>5.6</td>
<td>Experiments on task based budget allocation</td>
<td>54</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Experimental evaluation of budget allocation</td>
<td>55</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Synthetic data</td>
<td>55</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Medical dataset</td>
<td>60</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Weather sentiment dataset</td>
<td>60</td>
</tr>
<tr>
<td>5.6.5</td>
<td>Sport dataset</td>
<td>61</td>
</tr>
<tr>
<td>5.6.6</td>
<td>Conclusions on task based budget allocation</td>
<td>62</td>
</tr>
</tbody>
</table>

6 Conclusions 65

6.1 Summary of results 65

6.2 Future work 66

Bibliography 69
Chapter 1

Introduction

1.1 Motivation

In the era of Big Data and Machine Learning there is an increasing necessity to deal with substantial amount of data. Machine Learning algorithms have the ability to predict and suggest some information by finding patterns in historical data. However in many problems, like image classification or test categorization, datasets miss the ground truth data and hence it results hard to extract some useful information from them. Given the large scale of the problem, it is also often inconceivable to manually input the ground truth with just the help of a limited group of experts.

This is where crowdsourcing comes into play; the system leverages on a big crowd of paid workers spread across the world to fulfil a task which cannot be solved by a machine at the current state of the art. Many on-line platforms allow to do this, such as Amazon Mechanical Turk [1] or Crowdflower [6]. In these systems a requester posts a job composed by many tasks and establishes a monetary reward for the completion of a task; then other users of the platforms, identified as workers, complete some of these tasks and receive their payment in exchange.

There are many popular applications of crowdsourcing. One example is text categorization, that means classifying documents to some topics (such as economy, sport, science, etc.). Another one is object recognition: an image is shown to the user and she must tell if some specific objects appear in it. One more application is sentiment analysis, where the crowd is required to tell whether some sentences express a positive or a negative feeling.

In some occasions the tasks may be very easy to solve; in others they can otherwise be relatively hard. In addition, some people might have a more appropriate skill set then others with respect to a specific problem. This
could be related for instance to the geographical origin of the contributors or to their age or gender. This is why the designer of the experiment can usually restrict the pool of workers by requiring that they meet some specific prerequisites. It is hence clear that in many cases designing the experiment in one way or another can have a big impact on the results, therefore each different experiment should be devised carefully. However, it is not always possible to isolate the ideal set of workers a priori. Within a crowd of reliable users, who meet all the prerequisites, there can still people more skilled than others. In general their expertise is uncertain, even though they are most likely less reliable than some carefully chosen experts. A standard practice to face this problem is the use of redundancy, meaning that every question is asked multiple times to different workers. A consequent matter, which is also a main topic of this thesis, is how to aggregate these answers in order to predict the true answer. This problem is known as label aggregation and is pretty crucial as it affects the quality of the result; many solutions have been proposed and we will go through them in the next sessions. The simplest approach is known as Majority Voting (MV) which just takes the most popular answer as the correct one. It may lack in accuracy but it is very simple and fast, and hence still popular. One drawback is that it does not consider the ability of the workers, therefore spammers and highly talented users are treated in the same way. The distinction between different types of workers is a crucial aspect of the problem. The type can be related to his ability, his country of origin, his education, background etc. It is hence possible to gather users of the same type under the same group or community. Workers within the same group typically share a common attitude towards the task and therefore their answers are correlated. As discussed in [29] the correlation between users may also be due to some asymmetry in how the information is retrieved from the crowd. For instance querying workers coming from distinct geographical areas would constitute an asymmetry, thus implying the creation of demography-based groups. Another aspect that may bias a worker’s behaviour is the access to different information sources: an experiment may require the users to express their judgement on some hard topic, while allowing them to conduct some researches using specific sources. Some may be asked to look on some on-line websites, while others to check an encyclopaedia. These two groups of people would thus probably be internally correlated. The key point is that from the experimenter point of view being aware of these communities can remarkably improve the quality of the aggregation. The challenge is how to find the groups of correlated workers when they are not known in advance. This problem, which we will refer to as group discovering, can be seen as a clustering problem over the set of
1.1. Motivation

Having a very accurate tool to perform label aggregation may immediately trigger a different kind of reasoning: by using a good algorithm we probably need less redundancy, therefore we can achieve good predictions by asking the same questions a smaller number of times. This translates into the problem of smartly distributing a task to the crowd under some budget constraints, known as budget allocation. In fact, although the individual worker payment is low, multiplying it by the number of tasks and by the redundancy level can yield high costs, hence optimizing the budget allocation can bring a significant benefit. Throughout this work we are also going to deal with this subject. Quality assurance and budget allocation may be seen as two separate challenges, but in many cases they end up exploiting the same model.

In any case to address both the aggregation and the allocation problems we are going to rely on the worker grouping. Hence this matter acquires substantial importance since its resolution would help tackling the other two challenges.

The use of communities is not the only expedient that one can think of to improve the prediction. Many experiments are composed by a heterogeneous set of tasks. This means that tasks can be grouped into different categories, and those with the same category are all similar to each other. There are many experiments which involve heterogeneous jobs, however their spreading has been limited by the lack of an appropriate crowdsourcing platform. In fact today’s platforms assume that tasks are homogeneous and do not provide any tool to mark a distinction between them or to exploit this aspect in order to improve the quality of the outcome. Being aware of the task type could indeed constitute a big advantage. The behavior of workers as well as their abilities may be different depending on this type.

To make an example which can clarify how a heterogeneous experiment would be, we consider a real world dataset which is composed by many images of famous people [15]. Recently developed machine learning algorithms can successfully recognize people from their pictures. This dataset is used as training data to create an algorithm capable of naming celebrities by looking at their photo. Since the dataset is composed by one million people, the experimenter could decide to acquire the ground truth to train the algorithm by crowdsourcing the data. Contributors would thus be required to assign the right name to the celebrity depicted in the photo. This experiment is a perfect example to understand what heterogeneous tasks
1. Introduction

Figure 1.1: Celebrities coming from cinema, music, politics and sport [12, 31, 32].

are. In fact celebrities can be famous in different areas; for instance the data could include people coming from politics, sport, cinema or music. Fig. 1.1 shows an example of celebrities in these areas. It is reasonable to assume that every contributor may have his area of expertise; some may be more expert on sport rather than cinema, some people may not be prepared in politics and so on. Knowing these properties beforehand could improve the quality of the aggregate answer because we would be able to remove the biases. Moreover it would have really interesting developments in the budget allocation problem. In fact it would be possible to choose to which worker a task should be assigned not only basing on the general precision of that worker, but more specifically basing on his ability to solve that exact kind of task.

Our contributions are hence not limited to the group discovery problem, but concern also the topic of quality assurance as related to the task category.

1.2 Overview

In the remainder of this chapter we are going to formalize the problems we are tackling and to introduce the notation which we will consistently adopt throughout the work.

Chapter 2 presents some related work on the field of crowdsourcing. It also introduces the Access Path Model (APM) [29] and the CorEx algorithm [37], two important tools which we are going to extensively use.

Chapter 3 addresses the problem of group discovering. Here we propose a novel approach to find groups of correlated workers by analysing the matrix of their contributions. One of the biggest challenges we will need to face comes from the sparsity of this matrix; we hence propose a technique which accounts for that.
Chapter 4 addresses the problems of label aggregation and budget allocation and we introduce two novel techniques which take into account the heterogeneity of the tasks. For the first problem our goal is to improve the accuracy of the aggregated answer by considering the type of task at prediction time, together with the rest of the available data. In the second problem we aim to find the right pool of workers to which assign a certain task by diversifying our strategy correspondingly to the task category.

Chapter 5 empirically evaluates our techniques by presentig experiments on various real world and synthetic datasets. Our datasets cover a broad range of problems that can be solved via crowdsourcing.

Finally Chapter 6 illustrates the final conclusions on our work together with some ideas for future work.

1.3 Problem Statement

Set-up

In this section we introduce the formalism that will be consistently used in the rest of the work. We suppose that a crowdsourcing experiment is composed by a group of tasks \( t_1, \ldots, t_n \). We model the true answer of a task with a random variable \( Y \) and the answers given by workers with a set of variables \( X = \{ X_1, \ldots, X_m \} \). We will consistently indicate by \( n \) the total number of tasks and by \( m \) the total amount of workers. The realization of a random variable is indicated by the lower case letter, e.g. the outcome of \( Y \) is indicated by \( y \). The set of possible values that \( Y \) can take is conventionally discrete and non negative, \( \mathcal{Y} \subset \mathbb{Z}_0^+ \). Likewise the \( X \) variables take values in the set \( \mathcal{X} \).

For the remainder of this work we will assume that we know the ground truth \( \{ y_1^*, \ldots, y_n^* \} \) for the tasks \( \{ t_1, \ldots, t_n \} \), which will be part of the training set. We will crowdsource these tasks anyway in order to collect data from the crowd. Once the algorithm has been trained, we can actually start using it on tasks \( t_{n+1}, t_{n+2}, \ldots \), for which we truly want to find out the answers. This is an assumption that we decided to make, although it does not necessarily need to hold. There are cases where retrieving a ground truth is hard or not doable; some algorithms are able to handle this eventuality and actually manage to learn the distribution of the outcome variable \( Y \) together with the rest of the parameters [11].
1. Introduction

As we said, tasks $t_1, \ldots, t_n$ are crowdsourced; the experimenter chooses a certain redundancy level, that is how many times the same question is given to the workers. When the experiment terminates, the collected data can be rearranged in the form of a matrix $X$ of size $n \times m$. $x_{ij}$ represents the answer of worker $j$ to task $i$. Whenever a task is not rated by a worker, we fill the cell of $X$ with the number $-1 \not \in \mathcal{X}$, or with any value not in $\mathcal{X}$. If the redundancy level is $r$ then every row of $X$ has $r$ known values.

The rating matrix $X$ together with the ground truth $y^* = \{y_1^*, \ldots, y_n^*\}$ is what constitutes the training dataset. In addition we can have the set of task categories $c = \{c_1, \ldots, c_n\}$.

As it was explained earlier with the example of the celebrity recognition experiment, we often consider the case where tasks belong to different categories. The category of a job is meant to mark a distinction between heterogeneous tasks belonging to the same experiment. If this is given, it will be represented with the random variable $C$ taking values in $C \subset \mathbb{Z}_0^+$. For example, in the case of the celebrity recognition, we would map the four categories politics, sport, cinema, music to four integer values in the set $\{0, 1, 2, 3\}$.

The information about the category of a task basically encodes its features and is known from the beginning. The value of $C$ is therefore always known and can be used to tailor the model so that it distinguishes between task types.

Problem formalization

Throughout this work we are going to tackle three different problems: label aggregation, budget allocation and group discovery.

The problem of label aggregation is the most studied and known in the field of crowdsourcing. It requires to find a smart way of aggregating the answers in order to produce an answer $\hat{y}$ which corresponds to the true one $y^*$ with the highest possible accuracy. The contribution we give comes from the direct utilization of the task category $c$ in the prediction.

The matter of budget allocation is also well known to the community. We assume that every worker $w_1, \ldots, w_m$ is associated with a cost $q = (q_1, \ldots, q_m)$. Let us indicate by $v = (v_1, \ldots, v_m)$ the indicator vector stating which workers took part in the evaluation of a task $i$:

$$v_j = \mathbb{I}[x_{ij} \neq -1] = \begin{cases} 1 & \text{if worker } w_j \text{ completed the task} \\ 0 & \text{otherwise.} \end{cases} \quad (1.1)$$

In the allocation problem we are given a budget $B$ which is the maximum amount that can be spent to solve a task; the total cost of the experiment
will therefore be at most $B \times n$, with $n$ is the number of tasks.

The problem can be stated as finding the best vector $v$ such that $v \cdot c \leq B$. Here the definition of best vector is intentionally pretty vague, because its interpretation is left to the researcher. It usually means that we want to maximize the accuracy of the prediction or a related information measure.

The problem of group discovery is less common, but, as it will be clear in the following, its solution will open the way to the resolution of the other two problems. In fact being aware of the groups of workers allows us to use the access path model to perform inference and budget allocation. Hence, although we treat the three of them as distinct, they are intrinsically connected. The group finding can be expressed as a clustering problem over the columns of $X$. We will indicate by $l$ the total number of clusters and by $g = (g_1, \ldots, g_m)$ the assignment vector, where $g_j \in \{0, \ldots, l - 1\}$ and $g_j = k$ if $w_j$ is assigned to the $k$th community.

One big challenge in this setting is given by the sparseness of the matrix. Hence basic clustering algorithms are not directly applicable.
Chapter 2

Related work and background

2.1 Related work

Quality assurance

The problem of quality assurance in crowdsourcing has been already widely studied. Several methods and algorithms have been developed; the goal is to ensure a high accuracy of the prediction in the presence of noise and uncertainty. These two factors derive from the fact that not all the users answer correctly and that some of them intentionally spam the results. Their aim in this case it to earn the monetary reward without wasting time by worrying about giving the correct answer.

One first way to improve the prediction is to take the worker reliability into account. One pioneering work in this area comes from Dawid and Skene [11]. They introduced the EM algorithm to perform inference on their model, which takes into account the individual biases of workers towards questions. This work became a reference point for consecutive researchers [18, 30, 40]. Bachrach et al. developed a probabilistic model that takes into account both the estimated difficulties of the questions and the abilities of the users to aggregate the answers and adaptively allocate resources [10].

Groups of workers

Later works highlighted how within the same crowdsourcing experiment there may be contributors of different types. In general the notion of type refers to the ability of a forecaster to provide correct answers, thus this concept is mostly related to the error rate of the user. Lamberson and Page in [24] find the optimal proportion of different types of workers. One important takeaway of their work is that even though one group may be better than another, the optimal strategy in the case of high redundancy uses answers from both the groups, rather than discarding completely the contri-
2. Related work and background

The idea of diversity as an important element to ensure high quality aggregations has appeared in several works and various disciplines [16]. In crowdsourcing it owes its first appearance to the book published by Surowiecki [33]. Venanzi et al. also rely on the distinction between worker types, using the terminology of community [36]. The work presents the Community Bayesian Combination Classifier (CBCC), a model that learns at the same time the confusion matrix of each community and the memberships of the users.

A bayesian model is developed also by Nushi et al. [29]. Their work expands the concept of related workers by introducing the notion of access path. The membership of a user to a certain access path is not necessarily due to the correlation arose in their answers, but can also be imposed by the experimenter himself. This model constitutes an important starting point for our work and will hence be thoroughly illustrated in Sec. 2.2.1.

Budget allocation

The problem of cost optimization in the context of crowdsourcing is typically equivalent to finding a smart allocation of a given budget. The goal is to obtain a good prediction even if the amount of answers taking part in the aggregation process is smaller. The problem has been widely addressed [27, 41]. It goes under the category of active learning, because the decision on which task to assign to a user while respecting the budget constraints must be taken on-line. In the context of active learning, the goal is to select only the most informative labels to train an algorithm in order to reduce the total cost, which derives from individual label acquisitions. One important difference with the general formulation is that in crowdsourcing the error on the acquired labels can be very high because these are not provided by experts. Hence selecting those with the least amount of noise is the core part of the challenge. Karger et al. propose in [20] a solution based on random graph generation; this is strictly bound to the inference algorithm which they then introduce, and is based on belief propagation. Their work is generalized by Vaughan and Wortman who account for different types of tasks and frame the problem as an integer program [35].

The Access Path Model presented in [29] can be also used as a starting point to choose the most informative workers.

Task-based models

We already introduced the concept of heterogeneous experiments: the tasks presented to the crowd can differ from each other in terms of difficulty, topic, duration, etc. We also discussed how the current state of the art platforms do not allow to treat tasks belonging to different categories in different ways. However the research has already begun investigating this topic. The work
presented in [19] addresses exactly this problem and proposes a graphical model capable of aggregating the labels from the crowd while taking into account the features of the task, specified a priori. Whitehill et al. present in [40] a probabilistic model which considers also the difficulty of the task. In other works [14, 38] the same problem is tackled making use of graph-based model, rather than probabilistic.

In general the goal of the research is to model the dependencies between the worker answers and the difficulty or type of task. Learning how the error rates of the crowd vary in response to the different features of the task can potentially bring high benefits both to the aggregation process and to the active learning problem.

2.2 Background

2.2.1 Access Path model

Before we start exposing the core of the thesis, it is necessary to summarize a model presented in the paper [29] by Nushi et al., which will be our starting point. For any detailed information you can refer directly to the paper; however since we are going to extensively use some of the tools presented in it, we need to first report some core concepts.

The paper introduces the Access Path Model (APM), a Bayesian probabilistic graphical model which is used as a tool to aggregate the answers of the crowd and to perform budget allocation.

The answers of the workers are modelled by the $X$ variables, whereas the true outcome is the $Y$ variable. Every task constitutes an i.i.d. sample of the model. The parameters are unknown a priori and are estimated using the training data coming from tasks $t_1, \ldots, t_n$, as explained in section 1.3.

The Access Path Model can be seen as an improvement of the simpler Naive Bayes Model, which is represented in Fig. 2.1. The figure shows the true label $Y$ which generates the answers $X$ from the crowd. Here the assumption is rather simple: the answer of a worker depends only on the true outcome of a task and is conditionally independent of the other workers given the outcome.

The concept of access path is then introduced. As discussed in the introduction, an access path marks a distinction in the way information is retrieved from the crowd. For example, if the experiment is designed in such a way that the workers are geographically differentiated, that constitutes an access path, because this differentiation affects the answers. However throughout the rest of the work we will intend as an access path also simply a community or group of workers, not necessarily distinguished because of the
2. Related work and background

![Naive Bayes model](image1)

Figure 2.1: Naive Bayes model [29].

![Access Path Model - APM](image2)

Figure 2.2: Access Path Model - APM [29]. $S[k]$ indicates the number of workers belonging to access path $k$.

The experimenter’s choices.

The Access Path Model (Fig. 2.2) makes use of this concept by adding one layer of hidden variables $Z$ in the prediction: they represent the aggregated answer of a path. Then, rather than modelling the final outcome as the aggregation of the entire set of answers, this is inferred using the values of the hidden variables.

The model can thus be divided into three layers. The first one is composed by the usual outcome variable $Y$. The second layer is the hidden one; there is a $Z$ variable for every access path and they are dependent on $Y$. Moreover there exist a relation of conditional independence among them given the outcome $Y$. This condition reflects onto the different access paths and is a key assumption of the model. The third layer includes the individual answers of the participants. They are grouped according to the community they belong to.

One important feature of this model is that it enables an effective parameter sharing among the $X$ variables within the same access path. In fact, rather than having individual per-worker parameters, the conditional probability tables are shared among the $X$ variables within the same access path. This entails not only that the parameter estimation process is faster, but also that less training data is necessary to make a good estimation.
2.2. Background

The inference process is carried out in [29] with an Expectation Maximization (EM) algorithm. This handles the missing information about the hidden layer. Moreover the sparsity of the data matrix $X$ does not constitute a problem for this algorithm and the parameters sharing compensates for the missing data.

The APM has proven to be an effective model being able to outperform other state of the art algorithms. One core challenge which is left unresolved though, is how to cluster the users in case the access paths are not known a priori. Moreover there is no distinction between different task types, as they are all processed indistinctly. These problems will be addressed in the rest of the thesis.

2.2.2 CorEx

The challenge of finding the worker communities can be seen as a clustering problem over the matrix $X$, where we want to group the columns together. Although we are going to discuss this matter in the following sections, we now introduce CorEx (Correlation Explanation), an algorithm developed by Steen and Galstyan [37]. This in fact will be one of our core tools to address the problem.

Given some high dimensional data, the target of CorEx is to find some hidden variables being able to explain the correlation in the data. More formally, a set of random variables $X_1, \ldots, X_m$ is given. $G$ is a subset of indexes $G \subseteq \{1, \ldots, m\}$ and $X_G$ is the corresponding subset of random variables. The aim of the paper is to find $l$ hidden variables $Z_1, \ldots, Z_l$ such that they best explain the correlation of some groups $X_{G_1}, \ldots, X_{G_l}$, which are initially unknown. The groups $G_1, \ldots, G_l$ constitute a partition of the set $\{1, \ldots, m\}$, hence $\bigcup_{1 \leq k \leq l} G_k = \{1, \ldots, m\}$ and $G_k \cap G_h = \emptyset \forall k \neq h$.

Intuitively this means that the variables within a single group $G_k$ are correlated among each other and their correlation can be explained by the existence of a hidden variable $Z_k$. Fig. 2.3 shows the graphical model. One simplifying assumption is that a variable $Z_k$ is responsible only for the variables of its group $X_{G_k}$, hence two $X$ variables in two different groups are independent from each other. This condition may be too strong, so one could apply the same algorithm to the $Z$ variables and explain the correlation among them, thus adding one additional hidden layer on top.

To find the clusters of correlated variables, the total correlation $TC(X_{G_k}; Y_k)$, which is a measure of information gain, is maximized. More precisely, the total correlation of a group $G$ of variables is defined as:

$$TC(X_G) = \sum_{i \in G} H(X_i) - H(X_G),$$  \hspace{1cm} (2.1)
2. Related work and background

where \( H \) is the entropy function, while the total correlation of conditioned variables is:

\[
TC(X_G|Z) = \sum_{i \in G} H(X_i|Z) - H(X_G|Z). \tag{2.2}
\]

To measure how much \( Z \) explains \( X \) they use the notation with the semi-colon:

\[
TC(X;Z) = TC(X) - TC(X|Z). \tag{2.3}
\]

The optimization problem is then formulated as:

\[
\max_{G_k, p(z_k|X_{G_k})} \sum_{k=1}^l TC(X_{G_k}; Z_k) \quad \text{s.t. } |Z_k| = p, \ G_k \cap G_{k'} = \emptyset \tag{2.4}
\]

where \( p \) is the cardinality of the \( Z \) variables.

What CorEx does, is to solve this maximization problem. An indicator variable \( \alpha_{j,k} = \mathbb{I}[X_j \in G_k] \in \{0, 1\} \) is introduced with the constraint \( \sum_k \alpha_{j,k} = 1 \), which is the group non-overlapping condition. After some operations, the following property is derived, where \( x = (x_1, \ldots, x_m) \) is an occurrence of the joint probabilities of the \( X \) variables:

\[
p(z_k|x) = \frac{1}{Z(x)} p(z_k) \prod_{j=1}^m \left( \frac{p(z_k|x_j)}{p(z_k)} \right)^{\alpha_{j,k}}. \tag{2.5}
\]

To find the solution, the algorithm repeatedly updates the values of \( p(z_k|x) \) and \( \alpha_{j,k} \) by using Eq. 2.5 together with a formula defining the update step for \( \alpha_{j,k} \).

The authors also claim that although there is no guarantee that this procedure will find the global optimum, the objective is bounded by \( TC(X) \).

The paper then continues by illustrating the performance of CorEx, which turns out to be very good on the proposed datasets. The interested reader can find more information in [37].
The nice aspect about this algorithm is that the probabilistic structure that it infers resembles the one of the APM. One more interesting property is that it naturally handles missing data. Hence given an incomplete matrix $X$, where lines are observations, the program can cluster its columns following the criteria described before. A python implementation of the algorithm developed by the authors can be found on-line [5].

2.3 Greedy Budget Allocation

As discussed earlier in 2.1, there is an ample literature concerning smart budget allocation techniques. We now summarize the one proposed by Nushi et al. in the same paper cited before, where the APM is exposed [29].

After having illustrated the APM, the work introduces the notion of access plan, which defines how many people per access path should be queried and is represented as a vector $S \in \mathbb{Z}^l$. The problem of budget allocation, or access path selection, is here formulated as finding the best access plan which leads to an accurate prediction while respecting some budget constraint $B$. Here it is assumed that workers from the same access path share a common cost; hence if $q$ is the cost vector, the total cost for one task is given by the inner product $q \cdot S$.

For instance a cost vector may be $[1, 2, 3]$ for a model with three access paths; if the access plan is $[4, 2, 3]$ then 4 users from the first group are queried, 2 from the second and 3 from the third. The total cost for one task would then be 17.

In order to measure how good the choice of a certain access plan is, the authors use the information gain over the $Y$ variable given the access plan $S$: $IG(Y; S) = H(Y) - H(Y|S)$, where $H$ is the entropy function. This measures how much knowing the answers from the access plan reduces the entropy. The more informative the access plan $S$ is, the higher will be the information gain $IG(Y; S)$. Hence to find the best access plan this measure has to be maximized, under budget constraints.

$$S_{\text{best}} = \arg \max_S IG(Y; S) \quad \text{s.t.} \quad q \cdot S \leq B \quad (2.6)$$

Computing the information gain is a very expensive operation, as it involves iterating over all the possible realization of the labels in the plan. Thus a sampling approach is used to estimate it [23]. Also finding the right plan using an exhaustive search would be complex [21]; however the paper presents an efficient greedy algorithm that finds an approximate solution. This procedure is based on the submodularity of information gain, which means
that adding one vote to the plan when many votes have already been collected from the crowd is less informative than adding one when just a few have been retrieved. More formally, if we introduce the unit plan \( u \in \mathbb{Z}^l \) s.t. \( \exists k \text{ s.t. } u_k = 1, u_{k'} = 0 \forall k' \neq k \), the submodularity property for information gain translates into:

\[
IG(Y|S+u) - IG(Y|S) \geq IG(Y|S'+u) - IG(Y|S') \quad \forall S, S' : \|S\|_1 \leq \|S'\|_1.
\] (2.7)

The maximization of submodular monotonic functions is a well-known algorithm [28]. It is greedy in the sense that at every iteration it adds to the current plan the unit plan which maximizes the improvement in information gain.

For completeness we report the pseudo-code in Alg.1. We will refer again to this greedy budget allocation algorithm in the rest of the thesis. We included here the fundamental concepts; however paper [29] provides more detailed information that the interested reader can consult.

\begin{algorithm}
\caption{Greedy budget allocation.}
\begin{algorithmic}
\Function{allocate_budget}{B, q}
\State \( S_{\text{best}} \leftarrow 0 \)
\State \( b \leftarrow 0 \)
\While{\( \exists i \text{ s.t. } b \leq q_i \) do}
\State \( U_{\text{best}} \leftarrow 0 \)
\For{\( k \leftarrow 1 \) to \( l \)}\Comment{For each Access Path}
\State \( u \leftarrow \text{unit_plan}(k) \)
\If{\( q_k \leq B - b \)}\Comment{\( u_k = 1, u_{k'} = 0 \forall k' \neq k \)}
\State \( \Delta_{IG} \leftarrow IG(Y; S_{\text{best}} + u) - IG(Y; S_{\text{best}}) \)
\If{\( \frac{\Delta_{IG}}{q_k} > U_{\text{best}} \)}
\State \( U_{\text{best}} \leftarrow \frac{\Delta_{IG}}{q_k} \)
\State \( S_{\text{max}} \leftarrow S_{\text{best}} + u \)
\EndIf
\EndIf
\EndFor
\State \( S_{\text{best}} \leftarrow S_{\text{max}} \)
\State \( b \leftarrow \text{cost}(S_{\text{best}}) \)
\EndWhile
\State \Return{\( S_{\text{best}} \)}
\EndFunction
\end{algorithmic}
\end{algorithm}
Chapter 3

Access path discovery

The first challenge that we address is the discovery of the access paths. As we discussed already the Access Path Model achieves very good performances but assumes that the access paths are known a priori. We hence want to apply some clustering algorithm to find good worker groups which fit well for the APM. It must be underlined again that although we use the terminology “access path”, this is not strictly intended as a way to retrieve information, but rather as a grouping of the contributors.

We already introduced the CorEx algorithm in section 2.2. However we still have not highlighted its potential use for our purposes. The most interesting thing about it is that its probabilistic graphical model reflects the one of the APM. In fact the algorithm exploits correlation between variables due to the presence of hidden causes. We hence expect that CorEx and AMP perform well together.

3.1 Impact of data sparsity

The biggest problem we need to face is given by the sparseness of the data matrix $X$. In fact the sparsity level can sometimes be reasonable, e.g. around 70%, but also very high, e.g. 98%. This can constitute a big problem, as we are going to show in the following.

We ran some experiments to check how much CorEx is able to deal with missing data. We made use of the Python code provided by the authors [5].

3.1.1 CorEx with generated data

Firstly we tried with some generated data. We want to generate a data matrix $X$ by adhering to CorEx probabilistic model. We then test the algorithm
by clustering the columns and checking how good the cluster reconstruction is, after having added some sparseness.

Algorithm 2 Error evaluation on generated data.

**Initialize:** `sparseness_values`, mean parameters `\(p_{avg}\)`, number of hidden variables `l \leftarrow 3`, sample size `n \leftarrow 1000`, `NUM_REPS \leftarrow 100`  

`errors \leftarrow empty\ list`

**for all** `s` in `sparseness_values` **do**

- `error \leftarrow 0`
- **for** `1 \leq i \leq NUM_REPS` **do**
  - `S \leftarrow random\ int\ vector(l, \ max\ per\ component = 100)`  → **Vector of length** `l`, \(S_k = \text{size of access path } k\)
  - `g^* \leftarrow (0, \ldots, 0, \ldots, 0)` \(S_0\ \text{times}\)
  - \(1, \ldots, 1\) \(S_1\ \text{times}\)
  - \(l, \ldots, l\) \(S_l\ \text{times}\)
  - `X \leftarrow generate\ data(S, p_{avg})`  → **Using model in Fig.3.1**
  - `X \leftarrow sparsify\ data(X, s)`
  - `\hat{g} \leftarrow CorEx(X, l)`
  - `error \leftarrow error + misclassified\ elements(g^*, \hat{g})`
- **end for**

`error \leftarrow error \div NUM_REPS`

`errors \leftarrow\ error`

**end for**

**return** `errors`

The procedure we use to evaluate the error is sketched in Fig. 2. At every iteration we generate some data to then apply CorEx on top of it and we store the reconstruction error; to minimize the noise we repeat this several times and then we keep the average error as a representative.

The first step of each iteration is the initialization of the `S` vector, which contains the size of each worker cluster. Its values are randomly chosen from \(\mathcal{U}[1, 100]\); we do this because we do not want to bias the result by choosing specific sizes.

After the `S` vector has been initialized, the data for the matrix `X` is generated, following the model in Fig.3.1. For every sample, which corresponds to a row of `X`, some random binary values are assigned to variables `Z`’s, one for each cluster. Then, using the provided conditional probabilities, a set of child binary variables `X` is generated for every `Z`. The generation is carried out `n` times; this leads to a matrix `X` of size `n \times m` where `m = \sum_k S_k`. All the samples are i.i.d.

The parameters of the model are chosen every time by adding noise to the provided vector `\(p_{avg}\)`. The parameters are different at the worker level; for each worker `w` the parameter `i`, encoding the `i^{th}` dependency on the parent
3.1. Impact of data sparsity

variable, is chosen as $p_{w_i} = p_{avg_i} \pm 0.05$.

We repeated the experiment three times: once we used strong parameters, with really low error rates, once with medium ones, with higher error rates, and once weak, with high error rates. The intervals are reported in Table 3.1. After the generation, the matrix X is then made sparse by substituting

$$P(X_i = 0|Z_i = 0) \quad P(X_i = 0|Z_i = 1)$$

| Strong       | [0.75, 0.85] | [0.15, 0.25] |
| Medium       | [0.65, 0.75] | [0.25, 0.35]  |
| Weak         | [0.55, 0.65] | [0.35, 0.45]  |

Table 3.1: Intervals used to sample parameters in the CorEx error evaluation experiment.

some randomly chosen entries with the value $-1$. The choice of the values to remove follows a uniform distribution. The sparse matrix is then fed into the CorEx algorithm which returns a clustering of the columns in the form of a vector $\hat{g} = [\hat{g}_1, \hat{g}_2, \ldots, \hat{g}_m]$ with $\hat{g}_j \in \{0, 1, \ldots, I\}$. This is compared to the true cluster vector for each permutation of the assignment indexes and the lowest error is retained. We need to consider every permutation because two clusters may group together the same elements but still assign different labels to every group. The error for a certain permutation is computed as:

$$\text{error} = \frac{\sum_{1 \leq j \leq m} \delta[g_j^* - \hat{g}_j]}{m} \quad (3.1)$$

where $g_j^*$ is the true assignment of variable $X_j$ and $\delta$ is the Dirac delta:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3.2)$$

The whole procedure is repeated NUM_REPS times and the average error for every sparsity level is output.

We also need to have a baseline to compare against. We hence consider a random assignment $g_{\text{rand}}$ as the threshold after which the algorithm performs poorly, which is given by a random permutation of the true cluster vector.

Results

Fig. 3.2 shows the experiment results. The reconstruction works pretty well with low sparsity levels. However the error becomes much larger when there are too many missing values, regardless the strength of the parameters. Moreover the use of weak parameters makes the reconstruction much
3. Access path discovery

Figure 3.1: Structure of Correlation Explanation used to generate data.

Figure 3.2: CorEx clustering error on binary generated data.

We can notice also an interesting phenomenon. If we focus on the curve representing the experiment with strong parameters, we observe how the error remains pretty constant until the threshold of 0.7 sparsity to then reach its minimum around the value 0.8. After this point the error grows quickly. It is interesting that the reconstruction does not work in the best way when the matrix is complete, but rather when the data points are sampled. This effect could be due to the overlapping of the three clusters in the high dimensional space. Sampling only some points from these clusters may help to give a clearer distinction between them.

The same phenomenon is visible for the experiment on the medium parameters, but does not happen with the weak parameters, where the reconstruction is much harder.
3.1. Impact of data sparsity

One general conclusion we can draw is that when the amount of missing data is too high, then recovering the original clusters is not doable, regardless of the parameter strength. However the actual threshold where the reconstruction becomes infeasible strongly depends on the CPTs.

3.1.2 CorEx on the personality dataset

The second experiment was performed on a real world dataset, the “Big Five” personality dataset [7], which is the same used by CorEx’s authors in [37]. The dataset includes answers from many people taking a study where questions are meant to reveal one’s personality. A participant has to answer with a rate from 1 to 5 stating how strong he feels about some assertions. A psychological theory [13] claims that people can be represented using five personality traits. The questions are designed in such a way that each of them can reveal a specific trait. It should hence be possible to cluster the questions so that those who belong to the same trait are grouped together.

As already verified by Ver Steeg, Galstyan et al. the CorEx algorithm is able to perfectly recover the true clusters. This is a result which is not achieved by other state of the art clustering methods; the closest one in terms of accuracy is Independent Component Analysis [37].

However our aim is to test what happens when some data misses. We follow a procedure similar to the one described in Algorithm 2, with the only difference that the data is not generated. Results are shown in Fig.3.4.

Also in this case the error grows quickly after a certain point, which is around 80% sparseness.

3.1.3 Comparison with k-means clustering

We would like to compare the results of CorEx with the ones of some other clustering algorithm. The problem though is that it is not easy to find a com-
3. Access path discovery

Figure 3.4: The figure plots the clustering error made by CorEx algorithm on personality dataset versus the amount of sparsity.

A common method which accounts for missing data. We hence used a technique to impute the missing values based on correlation between variables [22,34], which we briefly explain here.

The correlation is computed with the Pearson metric over the set of common rates between two variables:

$$\rho(X_j, X_{j'}) = \frac{\sum_{i \in d_{j,j'}} (x_{i,j} - \bar{x}_j)(x_{i,j'} - \bar{x}_{j'})}{\sqrt{\sum_{i \in d_{j,j'}} (x_{i,j} - \bar{x}_j)^2} \sqrt{\sum_{i \in d_{j,j'}} (x_{i,j'} - \bar{x}_{j'})^2}}$$  \hspace{1cm} (3.3)

where $d_{j,j'}$ is the vector of all $i$'s such that $x_{i,j} \neq -1$ and $x_{i,j'} \neq -1$ and $\bar{x}_j$ is the mean of all known entries in column $j$. This quantity is useful to define a similarity measure between two variables:

$$s(X_j, X_{j'}) = s_{j,j'} = \frac{n_{j,j'}}{n_{j,j'} + \lambda \rho_{j,j'}}$$  \hspace{1cm} (3.4)

where $n_{j,j'} = \text{length}(d_{j,j'})$ and $\lambda$ is a smoothing parameter.

To fill in a missing entry $x_{i,j}$ of $X$, we take the $p$ variables $X_{j'} \neq j$ whose similarity with $X_j$ is the highest and whose entry for task $i$ is known. We then impute the missing value as a weighted average of the $p$ entries, using as weights the similarity values:

$$\hat{x}_{i,j} = \frac{\sum_{j' \in \mathcal{P}} s_{j,j'} x_{i,j'}}{\sum_{j' \in \mathcal{P}} s_{j,j'}}$$  \hspace{1cm} (3.5)

where $\mathcal{P}$ is the set of indexes of the $p$ variables.

By using this procedure it is possible to impute all the missing entries of the
3.1. Impact of data sparsity

matrix. After having done that, any clustering algorithm can be applied on the complete version of $X$.

We decided to use k-means as a baseline algorithm and we experimented both on generated data and on the personality dataset. In the case of the synthetic data, the procedure is the one sketched in Alg. 2, with the only difference that instead of using CorEx we used k-means together with the imputation.

Results

For the synthetic data the results are shown in Fig. 3.5. The curves are pretty similar to the ones resulting from the CorEx clustering; however the effect of the sparseness appears at lower levels.

The surprising results, which were observed also by the author of CorEx, are the ones concerning the personality dataset. As it can be observed in Fig. 3.6, the error is very high even with no missing values; moreover it scores worse than a random clustering after the threshold of 50% sparseness. The reason why this happen is that the sizes of the clusterings become very unbalanced and the data points are mainly grouped under one big cluster. In the case of a random assignment this does not happen so the error is lower.

![Figure 3.5: K-means clustering error on synthetic data.](image)

23
3. Access path discovery

![K-means error on personality dataset](image)

**Figure 3.6:** K-means clustering error on personality dataset.

### 3.1.4 Discussion about CorEx

The results presented so far have shown how CorEx manages to recover the true clusters even with rather sparse matrices. Its application to real world data has proven its advantage over the k-means algorithm. It is also particularly suited to be used in combination with the APM, because it models correlation between variables, rather than other similarity measures like the Euclidean distance. Moreover it naturally handles missing values, hence there is no necessity to impute them with additional techniques.

In Sec. 5.4.1 we will also test CorEx on data directly generated from the APM and we will have another confirmation of its usefulness.

We hence believe that this algorithm can be the key to find groups of correlated workers in crowdsourcing environments. However if too much data is missing, the algorithm hardly succeeds in identifying a good clustering.

It is pretty common in the field of crowdsourcing to deal with very sparse data matrices, whose percentage of missing values can easily exceed 90%. This can be the case when a lot of data is collected and few redundancy is applied. It is not so unlikely for this quantity to reach values like 98 - 99%. As the experiments shown, in these cases applying CorEx directly on the sparse matrices would not have any utility, because the assignment is basically performed randomly. The same holds for other traditional clustering methods such as k-means. This constitutes a major challenge to the problem of worker clustering.

To tackle this obstacle we elaborated a sparseness reduction technique which
basically reduces the size of the matrix. In the following we are going to present a solution that allows us to keep using CorEx even in condition of high sparsity.

### 3.2 Sparseness reduction via task grouping

We devised a technique to be applied to the datasets which have too many missing values, i.e. more than 80-90%.

We make use of the task category: as we stated already, we often consider the case where tasks are provided with a specific category or can be assigned one. Our assumption is that a worker should have a consistent behaviour within tasks of the same type. We are going to better formalize and test this hypothesis in Sec. 5.3.

**Algorithm 3**

Clustering of workers accounting for matrix sparseness.

```plaintext
function CLUSTER_WORKERS(matrix X, ground truth y*, categories c, number of clusters l)
    [n, m] = size(X) → Height and width of X
    if sparsity(X) < SPARSENESS_THRESHOLD then
        return CorEx(X, l) → CorEx returns the cluster indexes
    end if
    t ← \( \max_i c_i \) → We assume, without loss of generality, that \( C \subseteq C = \{1, \ldots, t\} \)
    \( X_s \) ← matrix of size \( t \times m \)
    for all \( 1 \leq c \leq t \) do → For each category c
        for all \( 0 \leq j < m \) do → For each column j
            \( \text{indexes} \leftarrow [i : c(i) = c \text{ and } X[i, j] \neq -1] \)
            \( x \leftarrow X[\text{indexes}, j] \) → Vector of all answers given by worker j within task category c
            if length(indexes) = 0 then
                \( X_s[c, j] \leftarrow -1 \)
            else
                \( e \leftarrow \text{error}(x - gt[\text{indexes}]) \) → Computes the error vector
                \( X_s[c, j] \leftarrow \frac{\sum_{i=1}^{\text{length}(e)} e^2}{\text{length}(e)} \) → Variance of worker’s error within cluster c
            end if
        end for
    end for
    \( X_s \leftarrow \text{discretize}(X_s) \) → CorEx only handles integer values
    return CorEx(X_s, l)
end function
```

Algorithm 3 depicts how worker grouping is performed and Fig. 3.7 exem-
3. Access path discovery

Figure 3.7: Sparsity reduction flow; the figure shows an example for a matrix with three workers and three categories.

plifies it. If the matrix $X$ has a sufficiently large amount of known values, then no special operation is needed and the clustering is performed simply using CorEx. Otherwise we build a second matrix $X_s$ where entry $(c, j)$ is the Mean Squared Error (MSE) made by worker $j$ within the tasks of category $c$. We basically assume that a user’s error does not vary too much if restricted to one specific task category; we will verify this in Sec. 5.3. This allows us to take the mean of the squared errors as a representative.

This operation will considerably reduce the sparsity of matrix $X$ while still providing a good starting point to apply CorEx. We do not expect that the clustering algorithm will find the same communities it would have found if applied to a less sparse version of $X$. In fact the variables of $X_s$ do not represent the answers, but rather a level of expertise. However the target is not necessarily to reconstruct the “true” access paths, but just to find some good communities which suit the Access Path Model. In this sense we will measure the quality of a clustering by looking at the improvement in accuracy.
3.2. Sparseness reduction via task grouping

As an important remark, we would like to point out that the error vector $e$ could take different forms. In the case where the task is a classification problem then the entries of the error $e$ for worker $w_j$ are defined as:

$$e_i = \begin{cases} 
0 & \text{if } x_{i,j} = y^*_j \\
1 & \text{otherwise.}
\end{cases} \quad (3.6)$$

If instead the task involves labelling some items with some score in a certain discrete range $\mathcal{X}$, then the bigger is the distance between the answer and the ground truth, the larger is the error made. Hence in this case $e$ is just defined as $y^* - x$. 
Chapter 4

Task category impact on aggregation and optimization

Crowdsourcing experiments involving heterogeneous tasks are pretty common. One example we already made, is the one of celebrity recognition, where the subjective task difficulty can vary based on the field (cinema, sport, politics or music).

In general we are addressing any experiment in which the tasks have always the same goal but present different features.

We think that taking the task category into account may be useful not only for the problem of label aggregation, but also for the one of budget allocation.

4.1 Jointly modeling task groups and access paths

We first tackle the matter of answer aggregation. We would like to introduce a modified version of the APM which jointly models the task category \( C \) together with the worker answers.

We thus decided to directly introduce the \( C \) variable in the Access Path Model, hence creating a modified version of it which we will refer to as Task Dependent Access Path Model (TD-APM).

One important matter is understanding how this variable affects the other variables in the model. Before exposing our final choice for the TD-APM, we are going to present some alternative models which all make use of the \( C \) variable.

One could think that the task category directly influences the answer of a worker, as shown in Fig. 4.1a. However this model does make use of the access paths since there is no distinction in the way \( C \) affects them, so it is not in line with the idea of the APM.
Another possibility is to connect the $C$ variable directly to the outcome $Y$, as represented in Fig. 4.1b. However, this would entail that the category directly affects the outcome of a task, which is clearly a very strong hypothesis, which hardly holds.

Given these considerations, we decided to build the TD-APM having the $C$ variable affecting the hidden variables $Z$, as Fig. 4.2 illustrates. In this way we encode the biases that a community may have towards certain kind of items.

To better understand how this would affect the prediction, we have to look at how the information flows from $C$ to $Y$. At aggregation time, the only known variables are $C$ and some of the $X$'s. The predicted label $\hat{y}$ is computed as:

$$\hat{y} = \arg \max_{y \in Y} p(Y = y | C = c, \bar{X} = x),$$

where $\bar{X}$ must be intended as the vector of known $X$'s. If we had no infor-

---

**Figure 4.1: Hypothesises for the TD-APM.**

(a) Here the $C$ variable directly affects the user labels.

(b) Here the $C$ variable directly affects the result.
4.2. Task-based budget allocation

The hypothesis stating that the error rates of the crowd may vary depending on the category of the task can be useful also to perform budget allocation. We hence would like to propose a task-based budget allocation strategy. Alg.1 of section 2.3 does not account for that.

One way to use the task category would be by maximizing the information gain conditioning not only on variables $X$'s, but also on the category $C$, by using the joint probability distribution deriving from the TD-AMP. One alternative way would be to use a completely different set of parameters for every task category $C$. This forces the parameters to be modelled around one single category type, thus encoding more strongly their dependency on the task.

We chose to use this strategy not only because the parameters are more
strongly conditioned by the type of task, but also because of the behaviour of the TD-APM in the prediction, which we are going to discuss in Sec. 5.5. The inference of different pools of parameters is achieved in a very simple way: instead of training the model on the whole dataset, we first subdivide this into partitions, one for each task type. Then we train individual models for every task kind and we derive optimal plans for them. This is achieved applying the greedy Alg. 1 on the inferred parameters. The final prediction is made using the global model, but the choice of the workers is made using the specific access plan for that task category.

Algorithm 4 This algorithm performs budget allocation taking the task category into account. It returns a Map where to every category value is associated a specific Access Plan.

```plaintext
function Task-based budget allocation(matrix X, ground truth y*, categories c, clusters g, budget B, costs q)
    result ← new empty Map
    for all cat in categories do
        indexes ← [i : categories[i] = cat] → Vector of indexes
        X_p ← X[indexes, : ] → We retain only the rows of X in indexes
        APM ← new APM(X_p, q, y*)
        APM.train()
        S ← find_optimal_plan(AMP, B) → Alg.1
        result[add] (cat, S)
    end for
    return result
end function
```

We expect this method to find better access plans in cases where users are strongly influenced by task types. However to aggregate the labels we still use the global access path model, i.e. the model trained on the whole dataset, without distinguishing between categories. The reason for this is that the parameter estimation of the global model is more precise because it makes use of more training data. So although it is probably less informative for the budget allocation process, it is more accurate in the prediction step.
Chapter 5

Experiments

In this chapter we present the experimental evaluation of the work. We will show the behaviour of the group discovery technique introduced in Sec. 3, together with the sparseness related problems. Next we present the performance of the TD-APM in the label aggregation process and the results of the task-dependent budget allocation.

5.1 Datasets overview

We firstly give a description of all the real world datasets we used for our experiments.

**Topic relevance dataset** [8] In this dataset the task is to tell whether a topic is relevant for a given document on a 3 level scale: 2 means highly relevant, 1 relevant and 0 non-relevant. There are 100 topics, which naturally constitute the task category variable, and 3261 documents evaluated by 722 workers, with a 99% sparsity for matrix $X$. On average every question is answered by 5.64 workers.

**Sport dataset** This was used in paper [29]; it is based on a betting competition\(^1\) were users had to predict the outcome of football matches. They have to provide the probability that a team wins, so their answers are expressed as a decimal value in the set $\mathcal{X} = [0,1] \subset \mathbb{Q}$. The ground truth is expressed as a binary variable, where 0 means that the first team won and 1 that the second team won. There are records for all the years from 2000 to 2007, however we consider data only from one year, 2003. The sparsity level for this year is very low; there are a total 256 matches with answers from 1960 people with an average redundancy of 1340 contributions per task, thus leading to a sparseness of just 33%.

\(^{1}\)www.probabilitysports.com
5. Experiments

We are given some additional information about each match, including the betting odds provided by bookmakers, which we are going to use as task categories: if according to the odds team 1 is stronger then $C = 0$, if they are about the same level then $C = 1$ and if team 1 is considered weaker then $C = 2$.

Medical dataset This data was collected by Nushi et al. in [29]. Users had to answer to detailed medical questions to which they are most likely unfamiliar with. Every question has a yes or no answer, so the data is binary. Some examples are: “is there any permanent cure for hepatitis C?”, or “can you diagnose Alzheimer’s from a urine test?”. Most people do not know the answers to this kind of questions, therefore they were allowed to consult some websites to retrieve the necessary information. However, since the goal of the paper was to test the APM, they were redirected to different information sources. This led to the creation of three access paths: a group of workers used websites containing answers from doctors, another group could read answers from patients and the last group used just some general forum.

The dataset contains 100 questions and 255 people took part to the experiment. On average 29 answers were collected per question, leading to a sparsity level of 89%.

The dataset by itself does not provide any task category variable. However it turned out that some questions were particularly easy whereas others were not. Hence we divided the tasks into two categories based on the agreement between workers.

Bluebird dataset [2, 39]. Users must tell whether an image contains a bluebird or not. It is a very simple classification task and all the data is binary. In addition the dataset is complete, so there is no sparsity. The number of tasks is 108, whereas the number of workers is 39. Unfortunately there is no task category variable.

Weather dataset [4] Workers must grade the sentiment of a certain tweets related to the weather; it can be positive, negative, neutral, unrelated or impossible to tell. The ground truth for this experiment was collected through another crowdsourcing job where contributors had to check the original sentiment evaluation. The matrix size is 1000 tweets × 102 workers with 20 answers per task and a sparsity level of 80%.

We could divide the tweets into three categories by looking at the words in them; those with negative words belong to category 0, those with positive words to category 1 and those with either none or both kind of words to category 2. Checking this type of words is a common practice in sentiment analysis [17]; the lists were retrieved from [9].

COCO dataset [3, 25] This contains a collection of images which were annotated through an advanced machine learning algorithm. This collec-
5.2 Code and libraries used

The community discovery procedure illustrated in Alg. 3 was written in Python. The reason for that is that it makes use of the open source implementation of CorEx [5].

The APM and the TD-APM were implemented in C# using the Infer.NET framework [26]. We used an inference engine which makes use of Gibbs sampling, which properly handles the missing information about the hidden variables.

To represent the model we made an extensive use of variable vectors; Fig. 5.1 shows how the models are structured in the framework.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size of X</th>
<th>Redundancy</th>
<th>Sparseness</th>
<th>Ground truth cardinality</th>
<th>Task category cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>3261 × 722</td>
<td>5.64</td>
<td>99.1%</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>Sport 2003</td>
<td>256 × 1960</td>
<td>1340</td>
<td>33%</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Medical</td>
<td>100 × 255</td>
<td>29</td>
<td>89%</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Bluebird</td>
<td>108 × 39</td>
<td>39</td>
<td>0%</td>
<td>2</td>
<td>×</td>
</tr>
<tr>
<td>Weather</td>
<td>1000 × 102</td>
<td>20</td>
<td>80%</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Coco</td>
<td>1000 × 278</td>
<td>5</td>
<td>98.2%</td>
<td>×</td>
<td>25</td>
</tr>
</tbody>
</table>

Table 5.1: Properties of used datasets.
Figure 5.1: Graphical representation of models in Infer.NET
5.3 Sparseness reduction

We first want to investigate the validity of the procedure exposed in Sec. 3.2, and more specifically in Alg. 3.

What we aim to show, is that it is reasonable to take the average squared error of a worker within a cluster. In this sense we expect a worker to be pretty much consistent when dealing with tasks of the same type, hence the variability of the error within a certain cluster should be low.

We took the datasets with the highest task category cardinalities, them being the topic dataset and the COCO dataset, and we looked at how the error made by the most hard-working contributors behaves within task clusters.

As explained in Alg. 3, if \( e_{j}^{c} = (e_{c1}^{j}, \ldots, e_{cE}^{j}) \) is the error made by worker \( w_{j} \) for all the items of type \( c \), the shrunk version \( X_{s} \) of matrix \( X \) will contain in position \((c, j)\) the mean squared error:

\[
X_{s}(c, j) = \frac{1}{E} \sum_{i=1}^{E} (e_{c1}^{j})^2.
\]  
(5.1)

Let us introduce the vector \( h_{j}^{c} = ((e_{c1}^{j})^2, \ldots, (e_{cE}^{j})^2) \), so that \( X_{s}(c, j) = \bar{h}_{c}^{j} \), the mean of \( h_{c}^{j} \). We also introduce the squared error made by worker \( w_{j} \) on the whole set of task, as the vector \( h_{j}^{i} = ((e_{1}^{j})^2, \ldots, (e_{E}^{j})^2) \); notice that in our notation we dropped the letter \( c \) identifying the task type. We expect the variance of vector \( h_{c}^{j} \) to be lower than the variance of vector \( h_{j}^{i} \); this would mean that within a specific task category the error is more stable than it is in general, hence authorizing us to use the average \( \bar{h}_{c}^{j} \) as an aggregated measure. This does not necessarily need to hold for all task categories; we will just expect that it is true on average:

\[
\frac{1}{|C|} \sum_{c \in C} \text{var}(h_{c}^{j}) \leq \text{var}(h_{j}^{i}),
\]  
(5.2)

where \( \text{var}(\cdot) \) is the population variance.

Example

To give a clarifying example, let us consider a worker \( w \) who contributed to 10 tasks in a hypothetical experiment where \( X = Y = \{0, 1, 2, 3, 4\} \). Let us also assume that the first 5 out of the 10 tasks belong to category \( C = 0 \), whereas the remaining 5 belong to category \( C = 1 \). Table 5.2 provides an example of these answers. As it can be noticed by observing the table, the error made by \( w \) is lower for tasks of category 0 than it is on those of category 1. This is in line with our assumption that a worker may be better at
5. Experiments

<table>
<thead>
<tr>
<th>Task</th>
<th>Category</th>
<th>Ground truth $y^*$</th>
<th>Answer $x$</th>
<th>Error</th>
<th>MSE</th>
<th>$\text{var}(h_c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.6</td>
<td>0.24</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>3.4</td>
<td>1.44</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Example of answers with different error rates among clusters.

Solving some types of tasks rather than others.

For the provided example, the vectors take the following values (we drop the superscript since we are considering only one worker): $e_0 = (1,0,-1,-1,0)$, $e_1 = (-2,2,-1,-2,2)$; $h_0 = (1,0,1,1,0)$, $h_1 = (4,4,1,4,4)$. The mean squared errors that we would use in $X_s$, the matrix of MSE’s, would hence be $h_0 = 0.6$ and $h_1 = 3.4$. What we claim is that it is reasonable to take these means because the variance of the $h_c$’s are low. In fact in this example $\text{var}(h_0) = 0.24$ and $\text{var}(h_1) = 1.44$ whereas $\text{var}(h) = 2.8$. The quantity $\frac{1}{|C|} \sum_{c \in C} \text{var}(h_c)$ is hence equal to $\frac{0.24 + 1.44}{2} = 0.84$ which is indeed lower than 2.8.

Results

Fig. 5.2 shows the result of this computation for the topic and the COCO dataset. For the latter we used the average of the answers as a ground truth; moreover the analysis is restricted to the answers concerning the global performance of the annotation engine, thus discarding other dimensions such as accuracy or common sense. The figures present the distribution of $\text{var}(h_c)$ for the 10 workers who gave the highest amount of answers. They also display the lines indicating the average variance of the squared error among clusters ($\frac{1}{|C|} \sum_{c \in C} \text{var}(h_c)$) and the variance of the global squared error ($\text{var}(h_c)$). As we can see Eq. 5.2 is satisfied in all cases and in most of them the gap is pretty wide, whereas in others is tighter.
5.3. Sparseness reduction

(a) Topic dataset.

(b) COCO dataset.

Figure 5.2: Variance of mean squared error for the ten most hard-working users. The histogram shows the distribution of \( \text{var}(h^i) \). The red dashed line indicates \( \text{var}(h^i) \) whereas the green solid one marks \( \frac{1}{|C|} \sum_{c \in C} \text{var}(h^i_c) \).
5. Experiments

5.4 Experiments on the group discovery algorithm

We would now like to test the performance of the group discovery technique (Alg.3). We hence first test it with some generated data to then try it on the real world datasets.

5.4.1 Synthetic data

The goal of this section is to test CorEx’s ability to discover the true clusters of workers when applied to synthetic data. We already tested CorEx achievements on generated data in Sec. 3.1.1, so we will not make very fine grained experiment and we will not spend too many words on this. The only difference with Sec. 3.1.1 is that now the generation process uses the entire AMP model, so the Z variables depend on the outcome of Y.

Our synthetic dataset uses binary variables and has three communities. We used pretty strong parameters, in the sense that their value is closer to 0 or 1 rather than to 0.5. Here is our choice:

\[
\begin{align*}
P(Y = 0) &= 0.500 \\
P(X_0 = 0 | Z_0 = 0) &= 0.179 & P(X_0 = 0 | Z_0 = 1) &= 0.882 \\
P(X_1 = 0 | Z_1 = 0) &= 0.091 & P(X_1 = 0 | Z_1 = 1) &= 0.873 \\
P(X_2 = 0 | Z_2 = 0) &= 0.162 & P(X_2 = 0 | Z_2 = 1) &= 0.923 \\
P(Z_0 = 0 | Y = 0) &= 0.093 & P(Z_0 = 0 | Y = 1) &= 0.768 \\
P(Z_1 = 0 | Y = 0) &= 0.247 & P(Z_1 = 0 | Y = 1) &= 0.889 \\
P(Z_2 = 0 | Y = 0) &= 0.099 & P(Z_2 = 0 | Y = 1) &= 0.869.
\end{align*}
\]

We hence produced a matrix X of size 1000 \times 90 by sampling the AMP; the sizes of the community are S = (25, 30, 35). We then removed 80% of the entries of X using an uniform distribution. This should simulate a matrix of answers in a crowdsourcing environment.

Results

We applied CorEx to the matrix X and the reconstruction was perfect: every column of X was assigned to the right cluster. We also led some experiments with slightly different parameters and sparsity thresholds and the result did not change.

It is thus clear that in reasonable conditions (i.e. good parameters and sparsity level, enough samples) this algorithm is perfectly suited to find the right communities.

However the optimal conditions are not always present; that is why we are now going to test CorEx on real world datasets.
5.4. Experiments on the group discovery algorithm

5.4.2 Group discovery performance evaluation strategy

We need a way to test how good a worker clustering is even if we do not have the ground truth for that, in the sense that we do not know the true groups. We consequently use a different strategy: we evaluate the error in the prediction using the APM. To this purpose we just make use of K-fold cross validation, as explained in Alg. 5.

We are always going to compare the result of this evaluation against some baselines. One is Majority Voting (MV): the predicted label is simply the answer with the highest number of appearances. We expect that APM with this worker clustering will perform better than MV. However to actually quantify how much of the improvement is due to the clustering, we are going to also use as a baseline the error of the APM when the worker clustering is random. This evaluation follows exactly the procedure sketched in Alg. 5 except for the fact that the cluster vector $g$ is randomly initialized.

**Algorithm 5** Error estimation for APM after communities have been discovered making use of CorEx. This procedure is based on K-fold cross validation.

```
Initialize num_folds
g ← CLUSTER_WORKERS(X, y*, c, l)  \rightarrow Alg.3
dataset ← new Dataset(X, y*, c, g)  \rightarrow Data structure representing a dataset
partitions ← split(dataset, num_folds)  \rightarrow List of datasets
error ← 0
for all partition ∈ partitions do  \rightarrow Cross validation
    partition_error ← 0
    testing ← partition
    training ← dataset \ testing
    model ← new APM(training)
    model.train()
    for all sample ∈ testing do
        ˆy ← model.predict(sample)
        if ˆy ≠ sample.y then
            partition_error ← partition_error + 1
        end if
    end for
    error ← error + (partition_error / partition.size())
end for
error ← error / num_folds
```
5. Experiments

5.4.3 Bluebird dataset

The first result we present concerns the bluebird dataset. The group discovery procedure does not need to shrink the data matrix when applied to the bluebird dataset, since there is no sparsity at all. Table 5.3 shows the settings of the experiment, which are the ones which achieved the best results among the ones we tried.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{X}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>$\mathcal{Z}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>$\mathcal{Y}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>3</td>
</tr>
<tr>
<td>num_folds</td>
<td>54</td>
</tr>
</tbody>
</table>

Table 5.3: Settings for the group discovery experiment on the bluebird dataset. We used 54 folds because they are half the number of samples.

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.24</td>
</tr>
<tr>
<td>APM with random worker groups</td>
<td>0.21</td>
</tr>
<tr>
<td>APM with CorEx clustering</td>
<td>0.13</td>
</tr>
</tbody>
</table>

Table 5.4: Results of the group discovery experiment on the bluebird dataset.

As we can see from Table 5.4 the use of APM in combination with CorEx outperforms both majority voting and the use of APM with some random assignment of workers. This last method performs still better than MV and as we are going to see this will be a pretty common behaviour. It is probably due to the fact that the complex structure of the APM works by itself better than the simple approach of MV.

5.4.4 Medical dataset

Regarding the medical dataset, since the sparseness is originally high, we use the matrix of mean squared error to perform the community finding, for which only 21% of the entries are missing.

The nice property about this dataset is that it already comes with some access paths. We can thus compare the achievements of our community discovery procedure to the scores coming from the pre-selected access paths.
5.4. Experiments on the group discovery algorithm

<table>
<thead>
<tr>
<th>$\mathcal{X}$ cardinality</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{Z}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>$\mathcal{Y}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>3</td>
</tr>
<tr>
<td>num_folds</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5.5: Settings for the group discovery experiment on the medical dataset.

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.25</td>
</tr>
<tr>
<td>APM with random worker groups</td>
<td>0.15</td>
</tr>
<tr>
<td>APM with pre-selected access paths</td>
<td>0.14</td>
</tr>
<tr>
<td>APM with CorEx clustering</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 5.6: Results of the group discovery experiment on the medical dataset.

Also in this case the CorEx based procedure works nicely, performing better than every other method.

5.4.5 Topic relevance dataset

Also for the topic relevance dataset we had to reduce the sparseness of the matrix before clustering. The cardinality of $\mathcal{X}$ and $\mathcal{Y}$ this time is 3, hence

<table>
<thead>
<tr>
<th>$\mathcal{X}$ cardinality</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{Z}$ cardinality</td>
<td>3</td>
</tr>
<tr>
<td>$\mathcal{Y}$ cardinality</td>
<td>3</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>3</td>
</tr>
<tr>
<td>num_folds</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 5.7: Settings for the group discovery experiment on the topic dataset.

we use the same of the hidden variables. In general having $|\mathcal{Z}| = |\mathcal{Y}|$ or $|\mathcal{Z}| = |\mathcal{X}|$ is the best solution.
5. Experiments

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.52</td>
</tr>
<tr>
<td>APM with random worker groups</td>
<td>0.43</td>
</tr>
<tr>
<td>APM with CorEx clustering</td>
<td>0.41</td>
</tr>
</tbody>
</table>

Table 5.8: Results of the group discovery experiment on the topic dataset.

Results

Again our strategy works better than the other methods, as shown in Table 5.8.

5.4.6 Weather sentiment dataset

Also for the weather dataset we reduce the sparsity by shrinking the matrix. Notice that this is a classification task, where \( X \) does not encode a range of scores, therefore the error is computed in accordance to Eq. 3.6. The cardinalities this time are set to 5, so estimating the parameters actually requires a substantial amount of data, but we have only two groups of workers.

<table>
<thead>
<tr>
<th>( X ) cardinality</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z ) cardinality</td>
<td>5</td>
</tr>
<tr>
<td>( Y ) cardinality</td>
<td>5</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>2</td>
</tr>
<tr>
<td>num_folds</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5.9: Settings for the group discovery experiment on the weather sentiment dataset.

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.028</td>
</tr>
<tr>
<td>APM with random worker groups</td>
<td>0.045</td>
</tr>
<tr>
<td>APM with CorEx clustering</td>
<td>0.035</td>
</tr>
</tbody>
</table>

Table 5.10: Results of the group discovery experiment on the weather sentiment dataset.

This time our algorithm performs slightly worse than MV; this is probably
due to the high redundancy and to the ease of the task. In fact also the errors are really low. We hence expect a better performance of the APM when the budget is limited. However the task grouping procedure achieves a good result.

5.4.7 Sport dataset

The peculiarities of the sport dataset are the high redundancy and the consequent low sparseness; we clustered the workers by applying CorEx directly on the original data. Since the answers $x_a$ of the crowd are in the interval $[0, 1] \subset \mathbb{Q}$ whereas our model only handles discrete values, we map the worker labels to the domain $\mathcal{X} = \{0, 1, 2\}$ simply using the function:

$$
\forall x \in \mathcal{X} \quad x = \text{mapping}(x_a) = \begin{cases} 
0 & \text{if } x_a \leq \frac{1}{3} \\
1 & \text{if } \frac{1}{3} < x_a \leq \frac{2}{3} \\
2 & \text{if } x_a > \frac{2}{3}
\end{cases}
$$

(5.3)

However just for the majority voting experiment we map them to the set $\mathcal{X}_{MV} = \{0, 1\}$, as it is reasonable.

<table>
<thead>
<tr>
<th>$\mathcal{X}$ cardinality</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{Z}$ cardinality</td>
<td>3</td>
</tr>
<tr>
<td>$\mathcal{Y}$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>3</td>
</tr>
<tr>
<td>num_folds</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 5.11: Settings for the group discovery experiment on the sport dataset.

Results

Table 5.12 shows the results of the experiment. APM’s scores are not much better than the others. This is most likely due to the very high redundancy of the dataset. In fact when many answers are provided it is not difficult

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.321</td>
</tr>
<tr>
<td>APM with random worker groups</td>
<td>0.328</td>
</tr>
<tr>
<td>APM with CorEx clustering</td>
<td>0.316</td>
</tr>
</tbody>
</table>

Table 5.12: Results of the group discovery experiment on the sport dataset.
5. Experiments

to aggregate them, and the most popular among them is probably the best choice.

5.4.8 Conclusions on the group discovery experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Absolute improvement against random assignment</th>
<th>Relative error reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bluebird</td>
<td>0.08</td>
<td>38.1%</td>
</tr>
<tr>
<td>Medical</td>
<td>0.03</td>
<td>20%</td>
</tr>
<tr>
<td>Topic</td>
<td>0.02</td>
<td>4.6%</td>
</tr>
<tr>
<td>Weather</td>
<td>0.01</td>
<td>22.2%</td>
</tr>
<tr>
<td>Sport</td>
<td>0.08</td>
<td>2.4%</td>
</tr>
<tr>
<td>Synthetic</td>
<td></td>
<td>0%</td>
</tr>
</tbody>
</table>

Cluster reconstruction error

Table 5.13: Summary of group discovery results.

As we illustrated so far the access path discovery algorithm works well in most of the cases. In particular, CorEx by itself is very good at recovering the original communities as shown with the synthetic data. Its application to real world crowdsourcing experiments has also proven to be really effective. Whenever the high sparsity does not allow to directly apply it, we can make use of the sparseness reduction technique to discover the access paths.

It appears to us that the performance of the clustering algorithm are highly dependent on the type of data it deals with. Excess of redundancy like in the sport and in the weather dataset make the Access Path Model less useful in comparison with simpler methods like Majority Voting. Therefore also an accurate choice of the access paths loses its importance. It is hence responsibility of the experimenter to apply the right tools for the experiments he has to deal with by distinguishing every specific case.

5.5 Experimental evaluation of TD-APM

We would now like to test how the novel Task Dependent Access Path Model performs in comparison to the traditional Access Path Model. The evaluation process is very simple: we first cluster the workers using the usual group discovery technique and then we estimate the prediction error through k-fold cross validation, as was done before.

We will apply this procedure both to some synthetic data and to real world datasets which include the category variable.
5.5. Experimental evaluation of TD-APM

5.5.1 Synthetic data

We generated a dataset of 1000 samples and 150 workers, grouped into 3 clusters of sizes [50, 55, 45]. Every sample is taken from a TD-APM with binary variables and a specific choice of parameters. We repeated the same experiments three times with different parameters sets: strong, medium, and weak. Their values is displayed in Figg. 5.14, 5.15, and 5.16. The strong set encodes ideal conditions: the error rates are generally pretty low (between 5% and 20%) and they also enforce strong biases with respect to the task category for each access path. For the medium set most of the error rates are between 15 and 25 % and not all the groups are biased by the type of tasks. In the weak environment the rates rise to the range 15 - 35 % and the influence of the variable C is marginal.

\[
P(Y = 0) = 0.500
\]
\[
P(C = 0) = 0.500
\]

| \(Z_0\) | \(P(X_0 = 0|Z_0)\) | \(P(X_1 = 0|Z_1)\) | \(P(X_2 = 0|Z_2)\) |
|---|---|---|---|
| 0 | 0.838 | 0.887 | 0.858 |
| 1 | 0.199 | 0.141 | 0.190 |

| \(C\) | \(P(Z_0 = 0|Y, C)\) | \(P(Z_1 = 0|Y, C)\) | \(P(Z_2 = 0|Y, C)\) |
|---|---|---|---|
| 0 | 0.901 | 0.903 | 0.511 |
| 1 | 0.080 | 0.122 | 0.498 |

Table 5.14: Strong CPTs used to generate synthetic data for the TD-APM experiment.

For each of the generated dataset we added some sparsity to simulate the real world answers. We used both a uniform and an exponential distribution for the missing values. The latter means that the number of missing entries per row follows an exponential distribution. This is done to simulate the behaviour of workers, who are rarely be really active and answer to many questions and are more often less productive and execute a smaller amount of tasks. To achieve this, for each column of \(X\), which corresponds
5. Experiments

\[ P(Y = 0) = 0.500 \]
\[ P(C = 0) = 0.500 \]

| Z_0 | 0 | 0.789 | | Z_1 | 0 | 0.840 | | Z_2 | 0 | 0.787 |
|-----|---|--------||-----|---|--------||-----|---|--------|
| 1   |   | 0.231  | |    | 1   | 0.165  | |    | 1   | 0.248  |

| Y   | 0  | 0.830 | 0.216 | | Y   | 0  | 0.819 | 0.158 |
|-----|----|-------|-------||-----|----|-------|-------|
| C   | 0  | 0.832 | 0.224 | | C   | 0  | 0.243 | 0.790 |

| Y   | 0  | 0.489 | 0.546 | | Y   | 0  | 0.815 | 0.202 |
|-----|----|-------|-------||-----|----|-------|-------|
| C   | 0  | 0.764 | 0.217 | | C   | 0  | 0.701 | 0.281 |

Table 5.15: Medium CPTs used to generate synthetic data for the TD-APM experiment.

\[ P(Y = 0) = 0.500 \]
\[ P(C = 0) = 0.500 \]

| Z_0 | 0  | 0.794 | | Z_1 | 0  | 0.792 | | Z_2 | 0  | 0.803 |
|-----|----|-------||-----|----|-------||-----|----|-------|
| 1   |   | 0.184  | |    | 1   | 0.203  | |    | 1   | 0.165  |

| Y   | 0  | 0.753 | 0.156 | | Y   | 0  | 0.815 | 0.202 |
|-----|----|-------|-------||-----|----|-------|-------|
| C   | 0  | 0.762 | 0.166 | | C   | 0  | 0.701 | 0.281 |

| Y   | 0  | 0.713 | 0.288 | | Y   | 0  | 0.829 | 0.280 |
|-----|----|-------|-------||-----|----|-------|-------|
| C   | 0  | 0.713 | 0.288 | | C   | 0  | 0.829 | 0.280 |

Table 5.16: Weak CPTs used to generate synthetic data for the TD-APM experiment.
to the contributions of a worker, we sample a number $D$ according to the exponential distribution:

$$D \sim \lambda e^{-\lambda x}.$$  \hfill (5.4)

We then retain only $\lceil D \rceil$ randomly chosen entries from the column of $X$. The parameter $\lambda$ is chosen in such a way to achieve the desired sparsity level, hence:

$$\lambda = \frac{1}{E[D]} = \frac{1}{(1 - s)n},$$  \hfill (5.5)

where $s$ is the percentage of missing values and $n$ the height of $X$. However the sparseness level was set to 80 and 90%, entailing an average redundancy of 30 and 15 answers per task.

We ran the same experiment with these different datasets, using the settings displayed in Table 5.17. We used as a benchmark the Majority Voting method, together with the usual APM and the Naive Bayes model.

<table>
<thead>
<tr>
<th>$X$ cardinality</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>$C$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>$Y$ cardinality</td>
<td>2</td>
</tr>
<tr>
<td>Access path sizes</td>
<td>[50, 55, 45]</td>
</tr>
<tr>
<td>num_folds</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5.17: Synthetic experiment settings.

Results

Table 5.18 shows the results of this experiment. As a general remark we notice that there is no big difference between exponential and uniform sparseness. We also notice that the TD-APM performs much better than all the other methods with strong and medium parameters. This is not true any more in the weak environment, where the CPTs have a very low dependency on the $C$ variable and have high error rates. In this case the scores of APM and TD-APM are very similar.

In general the errors are bigger with more sparseness, as expected. However the relative performances of the methods do not change.

5.5.2 Medical dataset

We tested the TD-AMP on the medical dataset. We did not have any real data directly modelling the task categories; however, as discussed already,
5. Experiments

<table>
<thead>
<tr>
<th>Strong parameters</th>
<th>80% sparsity</th>
<th>90% sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Exponential</td>
<td>Uniform</td>
</tr>
<tr>
<td>Majority Voting</td>
<td>0.567</td>
<td>0.543</td>
</tr>
<tr>
<td>APM</td>
<td>0.304</td>
<td>0.304</td>
</tr>
<tr>
<td>TD-APM</td>
<td>0.075</td>
<td>0.075</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.293</td>
<td>0.301</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Medium parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
</tr>
<tr>
<td>APM</td>
</tr>
<tr>
<td>TD-APM</td>
</tr>
<tr>
<td>Naive Bayes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weak parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
</tr>
<tr>
<td>APM</td>
</tr>
<tr>
<td>TD-APM</td>
</tr>
<tr>
<td>Naive Bayes</td>
</tr>
</tbody>
</table>

Table 5.18: Results of the TD-APM evaluation on the synthetic datasets.

there was a pretty natural distinction between easy and hard questions. To be more systematic, if $x_t$ is the vector of known answers for task $t$, then we assign categories in the following manner:

$$c(t) = \begin{cases} 
0 & \text{if } \text{var}(x_t) \leq 0.2 \\
1 & \text{otherwise.} 
\end{cases}$$ (5.6)

The cardinalities are hence the following: Although for this experiment we

| X cardinality | 2 |
| Z cardinality | 2 |
| C cardinality | 2 |
| Y cardinality | 2 |
| Number of access paths | 3 |
| num_folds | 20 |

Table 5.19: Settings for the TD-APM evaluation on the medical dataset.

would have the original access paths, we decided to group the workers using CorEx, since the performance was good.
5.5. Experimental evaluation of TD-APM

Results

The results of the experiment are displayed in Table 5.24. The score of the

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.25</td>
</tr>
<tr>
<td>APM</td>
<td>0.12</td>
</tr>
<tr>
<td>TD-APM</td>
<td>0.16</td>
</tr>
</tbody>
</table>

Table 5.20: Results of the TD-APM evaluation on the medical dataset.

TD-APM is pretty poor when compared to the usual APM. We tried to change the cardinality of $Z$ so that the model would have more states and could represent better the different combinations of the $Y$ and $C$ variables; however the results did not improve.

5.5.3 Topic relevance dataset

For the topic relevance dataset the distinction between tasks is well defined, so we expect better results from the TD-APM.

<table>
<thead>
<tr>
<th>$X$ cardinality</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z$ cardinality</td>
<td>3</td>
</tr>
<tr>
<td>$C$ cardinality</td>
<td>100</td>
</tr>
<tr>
<td>$Y$ cardinality</td>
<td>3</td>
</tr>
<tr>
<td>Number of access paths</td>
<td>3</td>
</tr>
<tr>
<td>num_folds</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 5.21: Settings for the TD-APM evaluation on the topic dataset.

Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.52</td>
</tr>
<tr>
<td>APM</td>
<td>0.41</td>
</tr>
<tr>
<td>TD-APM</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Table 5.22: Results of the TD-APM evaluation on the topic dataset.

The performances of the APM and the TD-APM are pretty similar with a
slight advantage of the APM.

5.5.4 Weather dataset

For the weather dataset we distinguished three type of tweets: those which contain negative words, those with positive words and the mix type, which either contain both negative and positive tokens ore none. We hence have three categories; the experiment settings are illustrated here:

| X cardinality | 5 |
| Z cardinality | 5 |
| C cardinality | 3 |
| Y cardinality | 5 |
| Number of access paths | 2 |
| num_folds | 20 |

Table 5.23: Settings for the TD-APM evaluation on the weather sentiment dataset.

Results

The results of the experiment are again disappointing.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.028</td>
</tr>
<tr>
<td>APM</td>
<td>0.043</td>
</tr>
<tr>
<td>TD-APM with CorEx clustering</td>
<td>0.054</td>
</tr>
</tbody>
</table>

Table 5.24: Results of the TD-APM evaluation on the weather sentiment dataset.

5.5.5 Sport dataset

The sport dataset comes with the betting odds provided by the bookmakers before every match. They encode the assumed superiority of the first team over the second one as a decimal number in the interval \([-15, 15]\). We made this discrete by equally subdividing it and mapping its parts to the set \(\{0, 1, 2\}\).
5.5. Experimental evaluation of TD-APM

| \( \mathcal{X} \) cardinality | 3 |
| \( \mathcal{Z} \) cardinality | 3 |
| \( \mathcal{C} \) cardinality | 3 |
| \( \mathcal{Y} \) cardinality | 2 |
| Number of access paths | 3 |
| num_folds | 8 |

Table 5.25: Settings for the TD-APM evaluation on the sport dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority Voting</td>
<td>0.321</td>
</tr>
<tr>
<td>APM</td>
<td>0.328</td>
</tr>
<tr>
<td>TD-APM</td>
<td>0.390</td>
</tr>
</tbody>
</table>

Table 5.26: Results of the TD-APM evaluation on the sport dataset.

**Results**

The results highlight a much poorer performance of the TD-APM versus the other methods. Our conclusion was that the betting odds may be a poorer predictor that the one coming from the crowd contribution.

5.5.6 **Conclusions on the TD-APM**

This tables reports the summary of the experiments (for the synthetic data, only the results with missing values following the exponential distribution are reported). It is not straightforward to give a correct interpretations of

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Absolute improvement against APM</th>
<th>Relative error reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80%</td>
<td>90%</td>
</tr>
<tr>
<td>Synthetic strong</td>
<td>0.229</td>
<td>0.229</td>
</tr>
<tr>
<td>Synthetic medium</td>
<td>0.052</td>
<td>0.048</td>
</tr>
<tr>
<td>Synthetic weak</td>
<td>-0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>Medical</td>
<td>-0.04</td>
<td>-0.04</td>
</tr>
<tr>
<td>Topic</td>
<td>-0.01</td>
<td>-0.01</td>
</tr>
<tr>
<td>Weather</td>
<td>0.011</td>
<td>0.011</td>
</tr>
<tr>
<td>Sport</td>
<td>-0.062</td>
<td>-0.062</td>
</tr>
</tbody>
</table>

Table 5.27: Summary of experiments on TD-APM model.

the outcomes of the experiments on the TD-APM.
On one hand its achievements on the generated data highlighted a sensible advantage of this model over the usual APM. However, as it was to be expected, the advantage becomes weaker when the parameters leave room to uncertainty and lose their dependency on the task category.

On the other hand the TD-APM did not improve the accuracy of the prediction in any of the experiments that we carried out, and in some occasions it made it even worse. This might be due to several reasons. One first consideration is that one drawback of the TD-APM is that it exploits more parameter, which make use of a higher number of variable combinations. Hence inferring them may require a high amount of training data.

One additional possibility concerning our experiments is that the pool of datasets we used did not contain any with a strong dependency on the task type. In fact both for the medical and the weather dataset we were forced to manually introduce a category. Moreover the use of betting odds in the sport dataset was probably a bad choice because it does not respect the assumption of independence between the task category variable and the outcome variable. Therefore our model was probably not the right one for this type of data; in addition the prediction made by bookmakers may by itself be a bad predictor. The topic dataset was actually the most suited one for our purposes and with it the TD-APM achieves a score close to the one of the APM. This dataset in fact has a clear distinction between tasks from the beginning and the topic is independent from the outcome of the task. It must also be underlined that most of the crowdsourcing experiments that have been undertaken so far deal with homogeneous tasks because there is a lack of aggregation models that support heterogeneity.

Another possibility that would explain the poor results could be that the TD-APM does not model well the real world data. It may be that workers are not biased so much in their behaviour by the type of a task; it may also be that the topology of the model is not correct, hence it is not right to assume that the task variable has a direct influence on the hidden layer.

### 5.6 Experiments on task based budget allocation

So far we have experimented the group discovery technique and the TD-APM; we are now going to test the budget allocation strategy exposed in Sec. 4.2.

As before the first experiments we present make use of synthetic data, while the rest uses the pool of datasets that we already described.
5.6. Experiments on task based budget allocation

5.6.1 Experimental evaluation of budget allocation performances

We need a way to check how good a budget allocation choice is. The idea is simple: to test a certain access plan, we sample workers accordingly from each access path to then aggregate their answers and check the accuracy of the prediction. To better estimate the error by simulating a real world usage, we still make use of cross validation, thus training the model on a portion of the data and evaluating it on another.

Alg. 6 explains more in detail how the error is computed; it makes use of a budget B and a cost vector $q$, which states the individual cost of every access path. Some of them may be more expensive than others, as explained in [29]; however since our analysis is focused on the discovery of groups of workers on which we have no a priori knowledge, we only consider the case where all access paths have the same cost. We included the cost vector $q$ in the algorithm for completeness, but in all the experiments we are going to chose $q = [1, 1, \ldots, 1]$.

Some baselines are computed using the traditional budget allocation method (Alg. 1) and Majority Voting with random allocation. For the former we evaluate the error following Alg. 6 with the only difference that there is no distinction among categories. The sampling for MV is performed randomly: for every task we simply take a sample without replacement of length $B$ from the available answers.

5.6.2 Synthetic data

As usual we generated some synthetic data to experiment with the budget allocation procedure. In order to simulate the dependency between the crowd behaviour and the task category we generated the data using the TD-APM, which encodes this correlation. However for the budget allocation and the prediction we did not make use of it, but we used the traditional APM, exactly as stated in Alg. 6.

As for the synthetic data of Sec. 5.5.1, we generated some datasets of 1000 samples each, with three groups of workers of sizes $[50, 55, 45]$; the size of $X$ is hence $1000 \times 150$. Also this time we tried with different sets of parameters, a strong set, a medium one and a weak one. The parameters are displayed in Tables 5.28, 5.29 and 5.30.

These CPTs are different from the ones listed in Sec. 5.5.1 as they are chosen to ensure that a wide range of parameter combinations are covered. In particular the probabilities $P(Z_i|Y, C)$ vary a lot depending on $Y$ and $C$. The shift from strong to weak parameters is mainly evident on the error rates of
5. Experiments

\[ P(Y = 0) = 0.500 \]
\[ P(C = 0) = 0.500 \]

| \( Z_0 \) | \( P(X_0 = 0|Z_0) \) | \( P(X_1 = 0|Z_1) \) | \( P(X_2 = 0|Z_2) \) |
|---|---|---|---|
| 0 | 0.915 | 0.866 | 0.929 |
| 1 | 0.098 | 0.074 | 0.056 |

| \( Y \) | \( C \) | \( P(Z_0 = 0|Y, C) \) | \( P(Z_1 = 0|Y, C) \) | \( P(Z_2 = 0|Y, C) \) |
|---|---|---|---|---|
| 0 | 0 | 0.818 | 0 | 0.06 |
| 1 | 1 | 0.076 | 0 | 0.44 |

| \( Y \) | \( C \) | \( P(Z_1 = 0|Y, C) \) | \( P(Z_2 = 0|Y, C) \) |
|---|---|---|---|
| 0 | 0 | 0.42 | 0.87 |
| 1 | 1 | 0.05 | 0.10 |

Table 5.28: Strong CPTs used to generate synthetic data for the budget allocation experiment.

\[ P(Y = 0) = 0.500 \]
\[ P(C = 0) = 0.500 \]

| \( Z_0 \) | \( P(X_0 = 0|Z_0) \) | \( P(X_1 = 0|Z_1) \) | \( P(X_2 = 0|Z_2) \) |
|---|---|---|---|
| 0 | 0.786 | 0.873 | 0.793 |
| 1 | 0.191 | 0.124 | 0.213 |

| \( Y \) | \( C \) | \( P(Z_0 = 0|Y, C) \) | \( P(Z_1 = 0|Y, C) \) | \( P(Z_2 = 0|Y, C) \) |
|---|---|---|---|---|
| 0 | 0 | 0.713 | 0.412 | 0.10 |
| 1 | 0.155 | 0.180 | 0.716 | 0.180 |

| \( Y \) | \( C \) | \( P(Z_2 = 0|Y, C) \) |
|---|---|---|
| 0 | 0.100 | 0.441 |
| 1 | 0.027 | 0.120 |

Table 5.29: Medium CPTs used to generate synthetic data for the budget allocation experiment.
5.6. Experiments on task based budget allocation

Algorithm 6 This procedure computes the prediction error through cross validation after performing budget allocation.

**Initialize** budget $B$, costs $q$, iterations $\leftarrow 10$

partitions $\leftarrow$ split(dataset, num_folds)

error $\leftarrow 0$

for all partition $\in$ partitions do

  testing $\leftarrow$ partition

  training $\leftarrow$ dataset testing

  globalAPM $\leftarrow$ new APM(training)

  globalAPM.train()

  access_plans_map $\leftarrow$ task-based allocation(training, $B$, $q$) $\rightarrow$ Alg. 4

  e $\leftarrow 0$

  for i $\leftarrow$ 1 to iterations do $\rightarrow$ We repeat more times to reduce noise

    for all sample $\in$ testing do

      plan $\leftarrow$ access_plans_map.get(sample.get_category())

      sub_sample $\leftarrow$ plan.random_sample(sample) $\rightarrow$ Sample answers

      prediction $\leftarrow$ globalAPM.predict_outcome(sub_sample)

      if prediction $\neq$ sample.ground_truth then

        e++

      end if

    end for

  end for

  e $\leftarrow$ e / testing.size() / iterations

  error $\leftarrow$ error + e

end for

error $\leftarrow$ error / partitions.size()

the worker answers given the hidden variables.

As usual the matrix $X$ of each experiment was made sparse by removing some values; the distribution of the number of missing values per worker is both uniform and exponential. The sparsity threshold was set again to $80\%$, entailing an average redundancy of 30 answers per task.

We repeated Alg. 6 for several values of $B$: $(3, 6, 9, 12, 15, 18)$. We compared these results with the traditional budget allocation strategy (Alg. 1) and with plain Majority Voting.

**Results**

The results are represented in Figg. 5.3, 5.4 and 5.5 for the three sets of parameters. As expected the error increases for all methods as the parameters
5. Experiments

\[ P(Y = 0) = 0.500 \]
\[ P(C = 0) = 0.500 \]

| \( Z_0 \) | 0 | 0.741 | 1 | 0.243 |
| \( Z_1 \) | 0 | 0.725 | 1 | 0.298 |
| \( Z_2 \) | 0 | 0.796 | 1 | 0.296 |

| \( Y \) | \( Z_0 = 0 \) | \( Y \) | \( Z_1 = 0 \) | \( Y \) | \( Z_2 = 0 \) |
| \( C \) | \( 0 \) | 0.498 | 0.155 | \( 0 \) | 0.038 | 0.109 |
| \( 1 \) | 0.156 | 0.049 | \( 1 \) | 0.565 | 0.160 |

| \( C \) | \( 0 \) | 0.195 | 0.053 |
| \( 1 \) | 0.388 | 0.106 |

Table 5.30: Weak CPTs used to generate synthetic data for the budget allocation experiment.

introduce more uncertainty, thus the prediction accuracy in the medium environment is generally lower than in the strong one, and the same happens between weak and medium.

Both for the strong and medium settings the use of the task based allocation reduces the error substantially with respect to the traditional allocation. The gap between these two methods is pretty large when the budget is low but becomes smaller as the budget increases to eventually disappear. This behaviour was expected: as soon as the budget approaches the full condition (30 answers in this case), the allocation strategy loses its importance, because most of the crowd is queried anyway. It is also important to underline that the model used in the prediction is exactly the same, so it is normal that the outcome of the prediction does not change much with high budget.

The advantage of the task related allocation method vanishes in the experiment with weak CPTs; here the two curves are both below the Majority Voting threshold, but are entangled with each other and almost overlapping.

It is also possible to directly look at how the budget is distributed to see how the information about the category affects this. In particular we observed the allocation for the experiment with strong parameters and \( B = 6 \), which is
5.6. Experiments on task based budget allocation

Figure 5.3: Budget allocation error on synthetic data - strong parameters.

Figure 5.4: Budget allocation error on synthetic data - medium parameters.

displayed in Table 5.31. It is interesting to notice that when the information about the category is ignored, then the second group is the most queried whereas the third one receives the lowest budget. However the proportions change a lot when the allocation is task based. In the case of category 0, the second group receives less budget in favour of the first one. For category 1, the first group is queried much less than the others. However the average allocation between the two categories is very close to the traditional one.
5. Experiments

![Budget allocation - Synthetic data with weak parameters](image).

Figure 5.5: Budget allocation error on synthetic data - weak parameters.

<table>
<thead>
<tr>
<th>Method</th>
<th>Group 1</th>
<th>Group 2</th>
<th>Group 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional allocation</td>
<td>2.05</td>
<td>3.10</td>
<td>0.85</td>
</tr>
<tr>
<td>Task-based allocation $C = 0$</td>
<td>3.90</td>
<td>1.80</td>
<td>0.30</td>
</tr>
<tr>
<td>Task-based allocation $C = 1$</td>
<td>0.35</td>
<td>4.45</td>
<td>1.20</td>
</tr>
<tr>
<td>average</td>
<td>2.125</td>
<td>3.125</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 5.31: Average budget allocated to each access path in the experiment with synthetic data with strong parameters and total budget $B = 6$. The experiment was done with 20 folds, and these vectors are the averages over every fold. The last row shows the average between the budgets given to category 0 and 1.

5.6.3 Medical dataset

The experiment for the medical dataset was made with $B$ ranging from 2 to 18. Fig. 5.6 illustrates the outcome of the experiment. What is clearly observable is a lower error of the APMs than the one of MV. However the advantage of one budget allocation method is not clear: the curves are slightly wiggly and pretty entangled.

5.6.4 Weather sentiment dataset

Fig. 5.7 displays the results of the experiment on the weather sentiment dataset, where the budget varies in between 2 and 14, whereas the full budget is 20. The curves are more stable and a slight advantage of the task
5.6. Experiments on task based budget allocation

Figure 5.6: Budget allocation error - medical dataset. The errors with full budget are also displayed.

Figure 5.7: Budget allocation error - weather dataset. The errors with full budget are also displayed.

dependent budget allocation strategy is visible.

5.6.5 Sport dataset

The last experiment makes use of the sport dataset; the result is displayed in Fig. 5.8. The full budget results are not displayed due to the high answer.
5. Experiments

Experiments redundancy; however the reader can find them in Table 5.12. Also in this case the curves are slightly entangled, so there is no clear superiority of one budget allocation strategy over the other.

5.6.6 Conclusions on task based budget allocation

We would like to draw some final conclusions on the budget allocation experiments.

The results deriving from the synthetic data are really promising. With both strong and medium parameters the advantage of the task based allocation is evident. It has to be emphasized once again that although we used the terms strong and medium, these configuration are not modelling ideal situations; the strong CPTs are actually the typical scenario in terms of error rate of workers. So having good results for both this and for the medium data is a good achievement. Moreover we would like to draw the reader’s attention on the fact that it was not so obvious that the method would perform well with generated data. In fact the data was created from a different model than the one used both for the allocation strategy and for the prediction. The maximization of the information gain has proven to be effective on the data even if it was generated from a slightly different model.

However the experiments on the real world data were not as conclusive as the formers. The curves never favoured a strategy over the other except for
the medical dataset, where the gain given by the task based method is any-
way rather minute. Our hypothesis for the reasons behind these outcomes
are the same that we made to explain the TD-APM performances. It might
easily be that the dependency from the task category is low in these datasets;
that would explain why the graphics are similar to the synthetic data with
weak parameters.

One possibility that we did not test is the case of non uniform costs across
the access paths. The usage of the constant vector may smooth the differ-
ences between different budget allocation techniques because it privileges
the access paths which bring the highest information gain.
Chapter 6

Conclusions

6.1 Summary of results

In this work we have addressed three problems related to crowd sourcing: label aggregation, budget allocation and group discovery.

In our case the problem of group discovery owes its importance to the Access Path Model. This is in fact a powerful tool that can be used to tackle both the problem of label aggregation and the one of budget allocation. However to properly function it requires the workers to be clusters into different groups according to the correlation of their contributions. The way this model was initially presented made use of artificially created access paths; the experimenter would in fact group the contributors differently basing on the way information is retrieved.

To generalize the scope of this model in order to use it even if the access paths are not determined a priori, we proposed a clustering technique which makes use of the correlation among workers to assign them to different groups. One of the biggest challenges in this problem is given by the fact that the answer matrix is typically highly sparse. Most of the traditional clustering algorithms do not naturally handle sparseness; moreover trying to impute the missing values by leveraging on the correlation between variables is also a very hard task, given that computing this correlation needs a sufficient amount of common known entries.

We introduced a sparseness reduction technique which relies upon the coherent behaviour of workers when undertaking jobs of the same kind; this however requires that the tasks composing the dataset can somehow be assigned to different categories.

The clustering procedure has given good results both when applied to the original matrix of crowdsourcing data and when utilized with the reduced one of lower sparseness. To measure this improvement we checked the vari-
6. Conclusions

Evaluation of prediction accuracy made through the APM when different groups of workers were used.

Once we managed to successfully tackle the problem of group discovery, we tried to exploit the task category to refine the prediction of the APM and to improve the budget allocation strategy. Our starting point has been the APM in both cases, together with the rest of the work developed in [29].

To enhance the prediction capacity of the APM in cases where the crowdsourcing experiment is composed by heterogeneous tasks, we introduce the TD-APM, a modified version of the APM which includes the task category among its variables. The assumption behind the model is that some of the groups may have biased behaviours towards certain type of crowdsourcing jobs. The category variable has therefore an influence on the hidden layer which aggregates the answers of individual communities.

This new model has proven to achieve very good results in comparison with the APM when applied to synthetic data, even when the parameters used in the generation process encoded a high level of uncertainty. However the same model did not perform as well when applied to the real world datasets that we made ourselves available. Our main hypothesis to explain the result is that one of the assumption of the model was violated: it would not be true that certain group of workers were biased by the task category. This means that the task type has a low impact on the behaviour of the workers; it is likely that in the datasets we used the task differences played a marginal role.

The improvement on the budget allocation strategy has also given very good results on the synthetic data. The data was generated through the TD-APM but the allocation process and the inference part made use of the traditional Access Path Model. The distinction between tasks of different kind improved the quality of the prediction remarkably. However also in this case the results of the experiments on the other datasets were not as good; the accuracy of the prediction was not worsened but neither improved.

6.2 Future work

One direction that we did not explore in our work is the one of bi-clustering: if the task category is not known a priori, the researcher could try to cluster workers and tasks at the same time. However one big challenge in this case will be again the sparseness, which is the main reason why we did not investigate this area.
6.2. Future work

On the task based budget allocation and label aggregation there is still much room for improvement, given that our experiments did not have good performances. It would be interesting to design an experiment where the distinction between tasks is clear from the beginning and where this could highly affect the responses of the crowd. In these conditions the achievements of the TD-APM and of the task based budget allocation should be tested again.

It would also be interesting to see how the budget allocation varies in response to unbalanced costs of the access paths. This could entail that those with higher error rates are queried more often when their cost is low. It could be investigated how exploiting the knowledge on the task category would affect this allocation. Understanding this could have an immediate benefit as it would allow the requester to create different pricing schemes and remunerate more the groups with the highest accuracy. The advantage of the task-based allocation would be that the payment would change also according to the category of the task. Hence the same worker would receive different compensations depending on the task type. This translates in a more fine grained control from the requester’s point of view, who can assign a given task to the right community of workers and for the right price.
Bibliography


[32] Web Summit. Bono at the 2014 web summit. https://www.flickr.com/photos/websummit/15593861939/, November 2014. This image was originally posted to Flickr. The image was cropped and the overlay name was added.


[34] Matteo Pozzetti Svetoslav Karaivanov, Francesco Locatello. Relaxed item-based collaborative filtering. Paper written as part of Computational Intelligence Laboratory course at ETH, Zurich.


Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor’s thesis, Master’s thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Access Path Design for Quality Assurance in Crowdsourcing

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):
Pozzetti

First name(s):
Matteo

With my signature I confirm that

− I have committed none of the forms of plagiarism described in the ‘Citation etiquette’ information sheet.
− I have documented all methods, data and processes truthfully.
− I have not manipulated any data.
− I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date
Zurich, 05/09/2016

Signature(s)

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.