Hybrid Human-Machine Vision Systems: Image Annotation using Crowds, Experts and Machines

Thesis by

Peter Welinder

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2012

(Defended May 17, 2012)

To my family.

Acknowledgments

Choosing to pursue my Ph.D. at Caltech is one of the best decisions I have ever made. The beautiful setting in Pasadena, the knowledgable professors and fellow students, the amazing Vision-lab crowd, and the atmosphere of curiosity and learning have been incredibly inspiring for me.

The first person I'd like to thank is my adviser, Pietro Perona. I still remember his enthusiasm during an interview weekend as being one of the main factors that made me want to go to Caltech. He has the capacity to turn the inkling of an idea into something wonderfully deep and interesting. He has taught me to always think big, but to make sure I get the foundations right. Pietro's willingness to explore new ideas, both inside and outside the lab, has made my years at Caltech into a truly amazing adventure.

I really enjoyed the fun times I've had with the other members of the Vision Lab (in alphabetical order): Marco Andreetto, Ron Appel, Evgeniy Bart, Serge Belongie, Kristin Branson, Xavier Burgos-Artizzu, Krzysztof Chalupka, Bo Chen, Heiko Dankert, Piotr Dollar, Eyrun Eyjolfsdottir, Panna Felsen, Conchi Fernandez, Thomas Fuchs, Tyler Gibson, Ryan Gomes, Greg Griffin, David Hall, Ali Lashgari, Michael Maire, Yoav Schechner, Merrielle Spain. It's an amazingly smart and friendly group of people. In particular, I'd like to thank Piotr, who has been a great friend and inspiration, both inside and outside lab. I also had the pleasure of working with three extremely talented SURF students: Eirikur Thor Agustsson, Ilya Nepomnyashchiy, and Sara Venkatesh.

Much of my research wouldn't have happened without some amazing collaborators. I've had many fun moments with the Visipedia-team: Ron Appel, Boris

Babenko, Steve Branson, Serge Belongie, Florian Schroff, Catherine Wah. I also enjoyed Ryan Gomes and Andreas Krause's insights when working on the Crowd-clustering project, with Piotr Dollar on the pose-regression paper, and Max Welling on the performance estimation work. Finally, my side-projects with Jess Adkins and Simon Warby have been really interesting and fun.

I'd like to thank my candidacy and thesis committees for their advice and guidance: Yaser Abu-Mostafa, Jim Beck, Serge Belongie, Andreas Krause, Pietro Perona, Shinsuke Shimojo.

Lastly, I'd like to thank my family for their enduring support. Yana has been incredibly patient and supportive during my five years in grad school, and I look forward spending the rest of my life with her on new adventures. My parents and siblings have always encouraged my pursuits, and offered a wonderful and loving home to return to whenever I visited Sweden.

Abstract

The amount of digital image and video data keeps increasing at an ever-faster rate. While "big data" holds the promise of leading science to new discoveries, raw image data in itself is not of much use. In order to statistically analyze the data, it must be quantified and annotated. We argue that entirely automated methods are not accurate enough to annotate data in the short term. Crowdsourcing is an alternative that provides higher accuracy, but is too expensive to scale to millions of images. Instead, the solution is hybrid human-machine vision systems, where the work of both humans and machines is balanced to be as cost-effective and accurate as possible. With this goal in mind, we begin by categorizing different types of image annotations, and describe how nonexpert annotators can be trained to carry out challenging image annotation tasks. Having identified which types of annotations are appropriate for most tasks, including binary, confidence, pair-wise and continuous annotations, we present models for crowdsourcing annotations from hundreds of expert and nonexpert annotators (humans). By trading off the bias and expertise of multiple annotators, we show that it is possible to achieve high-quality annotations with very few labels. We show that the number of labels can be further reduced by actively choosing the best annotators to carry out most of the work. Finally, we study the problem of estimating the performance of automated classifiers (machines) used to annotate large datasets where few ground truth labels are available. Using a semisupervised model for classifier confidence scores, we show that it is possible to accurately estimate classifier performance with very few labels.

Contents

Chapter 1

Introduction

Over the last few decades, digital image sensors have gone from being a rarity to taking a fundamental place in many people's lives. For consumers, snapping a picture and sharing it with friends and family has never been easier. Most phones come equipped with one, if not multiple, cameras, and it is not difficult to imagine a near future where we will be recording every bit of our lives [BG09]. In industry, a growing number of satellites and unmanned aerial vehicles (UAVs) are mapping our planet at ever higher resolutions, surveillance cameras are appearing at every street corner, and doctors examine our bodies and tissues with high-resolution scanners and microscopes. Academic research labs across scientific disciplines, from astronomy to biology, are deploying new instruments capable of producing mind-boggling amounts of image data every day. Thus, across all domains, we are experiencing a deluge of image and video data, growing at an enormous pace. But raw "unstructured" data is not very useful in itself, if we do not have a way to understand it. The challenge is turning the raw data into "structured" data, that is, annotated data on which we can apply standard statistical and data mining methods.

While the amount of data has exploded, researchers in the fields of Machine Learning and Statistics have been busy developing many promising tools for dealing with "big data". General tools such as principal component analysis, matrix factorization techniques, random forests and support vector machines allow scientists and practitioners to make huge progress in everything from understanding gene and disease data to detecting fraud patterns and classifying text documents. Unfortunately, most

raw image and video data produced today is outside the capabilities of many popular statistical tools. The reason is that we lack ways to represent the images by their content, and the features that are available are still not much better than the raw pixels.

Turning pixels into quantified information is the goal of the field of Computer Vision. Though there is much interesting work left, the computer vision community has made significant progress over the last few years. Some technologies, such as face detection [VJ04] and object matching using SIFT features [Low04], have already made it into commercial products. Today, we can annotate faces and textured objects, such as books, movie posters, and art, with high accuracy in images. In science, computer vision is quickly becoming an indispensable tool for quantifying image and video data [DWH+09, FLW+08]. For some applications, it is possible to detect and track animals, so that their movements can be turned into location and velocity data, which in turn can be used to characterize behavior statistically [BRB+09]. However, the accuracies of most computer vision systems are not at the point where they can be reliably used for the broad range of data that is available today. To make today's methods work reliably, the experimental settings have to be constrained and large amounts of labeled data is required to train the computer vision algorithms. Even when trained with thousands of examples, the results need to be double-checked by experts, as the systems cannot be fully trusted.

Crowdsourcing has emerged a solution for the situations where computer vision is not yet accurate enough. The term, "crowdsourcing", was coined by Jeff Howe in an 2006 article [How06] and refers to the process of oursourcing small tasks to a large number of people. Since humans have highly developed visual systems, with years of training, we can quickly adapt to new visual tasks even with few training examples. The first crowdsourcing projects involved unpaid volunteers, with the best known example being the web-based encyclopedia, Wikipedia. Lately, scientists have realized the power of crowdsourcing, and numerous "citizen science" projects have been released to the public. Through the website Galaxy Zoo[1], astronomers enlist

[1] `http://www.galaxyzoo.org`

the help of amateurs to classify hundreds of thousands of photos of galaxies taken by the Hubble space telescope. The American space agency, NASA, asks citizen scientists to find craters on Mars by clicking on images on the planet surface through the "Be a Martian" website[2]. Archaeologists attempted (unsuccessfully) to crowdsource the search for the tomb of Genghis Khan by asking ordinary people to look through satellite images of Mongolian plains[3]. While crowdsourcing volunteer work is useful for large projects, online crowdsourcing markets such as Amazon Mechanical Turk[4] (MTurk), let "requesters" post tasks and bid for the time of "workers". Researchers in machine learning and computer vision have quickly adopted the tool to annotate large datasets [SF08, SOJN08].

The issue with today's crowdsourcing systems is that they have a lot of inefficiencies. To ensure high-quality annotations, multiple workers have to annotate the same image. This impacts the scalability of crowdsourcing, as annotating very large datasets can become prohibitively expensive. Moreover, some annotators are better at certain tasks than others (i.e., there are some "experts" in the crowd), so their time should be focused very carefully. There are also some tasks on which computer vision algorithms could possibly perform quite well, especially with a lot of training examples. For example, if the task is to count cells in tissue samples, a modern object detector could be trained to perform with quite high accuracy. Then, instead of annotating all the data, a human worker could simply aid the machine by correcting mistakes. Today, those tasks are still done by humans, even though they may be very mundane, impacting the overall motivation of the workers in the crowd. The challenge is to create a system where machines and humans can work together, doing the part of the work for which they are best suited.

[2] `http://beamartian.jpl.nasa.gov`
[3] `http://exploration.nationalgeographic.com`
[4] `http://www.mturk.com`

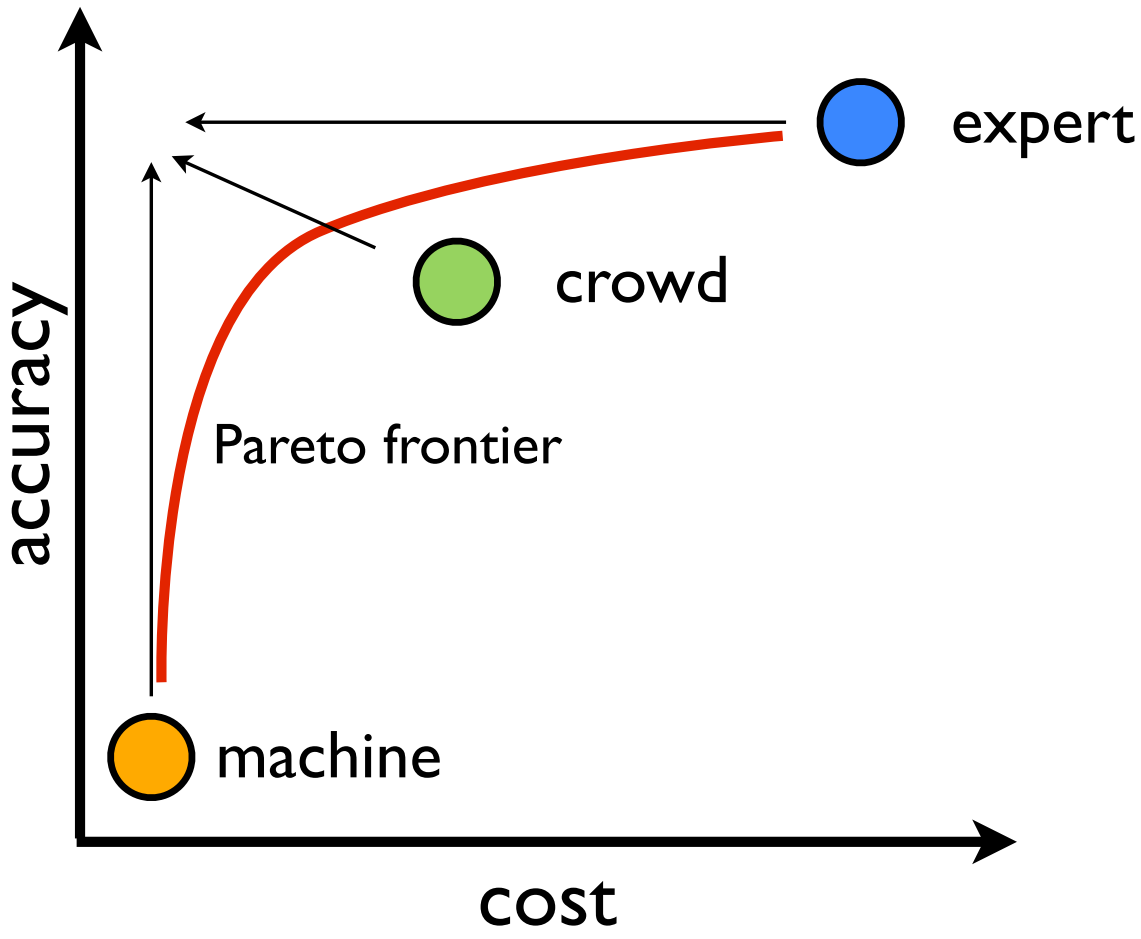

Figure 1.1: Tradeoff between cost and accuracy for hybrid human-machine vision systems. Machine algorithms have very low cost (basically computer CPU time cost), experts are very expensive but have maximum accuracy, and a "crowd" of non-experts fall somewhere in-between. The long-term vision is to develop a framework that trades off the strengths and weaknesses of each agent (machine, crowd, expert) along an optimal Pareto frontier. This goal of this thesis is to lay the foundation for that vision. The hope is that future research will push the frontier towards the upper-left part of the figure.

1.1 Hybrid Human-Machine Vision Systems

The goal of this thesis is to lay the foundations for humans and machines working together to annotate data. The vision is to build hybrid human-machine vision systems (HHMVS), which process images by balancing the work of humans and machines as cost-effectively as possible (see Figure 1.1).

There are three kinds of agents involved in HHMVS: (1) Experts are humans specifically trained at the task at hand, and so can make decisions better than anyone else. In many cases, it is the experts themselves that request the help of non-expert humans and machines, since their own time is very expensive. (2) A crowd of non-expert humans may have received brief training, but they do not have expertise in the task, and are often exposed to the images for the first time. (3) Machines (or automata) represent the automated computer vision methods that classify or detect objects in images. While machines are very specialized in what they can do, humans are quite flexible and can generalize well between tasks. For example, it is known that if trained with detailed instructions, human non-experts can perform at accuracy levels comparable to experts [BS87].

The questions considered in this thesis include: How do we characterize and measure the performance of human annotators when there is no ground truth? How can we use a crowd of noisy non-experts to annotate data and still guarantee high accuracy? What is the tradeoff between cost and performance in crowdsourcing? How does the accuracy of the crowd compare to experts? How can we estimate the performance of computer vision algorithms with as few expert-labeled examples as possible?

While the work in this thesis is focused on image data, many of the results can be applied to other modalities, such as video, time-series, audio, and text. For example, the work on binary annotation is very general, as many other annotation tasks can be broken down into multiple binary tasks.

1.2 Thesis Outline

This thesis is organized as a series of self-contained chapters, each discussing a different aspect of HHMVS. Some of the chapters have been published before, while others are still under submission at the time of writing. Related work in the literature is discussed separately in each chapter.

Chapter 2 describes different types of annotations and their use in applications and datasets. It also shows the interfaces for annotating images and for training annotators.

Chapter 3 (joint work with Steve Branson, Serge Belongie and Pietro Perona) presents a model for binary annotation. The model parameterizes annotator bias, competence, and expertise. By weighting the annotators based on the quality of the labels they provide, the method can be used to obtain high-quality binary labels. The experiments presented in the chapter show that all parameters are needed to ensure high accuracy. Finally, the model can be used to find experts in a crowd of annotators.

Chapter 4 (joint work with Pietro Perona) extends the work on binary annotations to polytomous annotations, or confidence scales. It presents theoretical results for how much information can be expected to be gained from confidence labels, and also proposes a method for training annotators to use confidence labels as efficiently as possible.

Chapter 5 (joint work with Ryan Gomes, Andreas Krause, and Pietro Perona) further extends the model in Chapter 3 to grouping images by visual similarity. The method is called "crowd clustering" and lets the annotators group images together based on loose instructions. By grouping images, the annotators effectively provide pairwise binary labels for the images. Using the model, it is possible to extract clusters of similar images, and to characterize annotators based on what features they are looking for. My main contribution to this work was developing and carrying out the experiments.

Chapter 6 (joint work with Pietro Perona) explores detection annotations, where

annotators are asked to detect objects in images. The experiments compare annotators on real and synthetic tasks, and show that the performance of non-expert annotators can be comparable to that of experts.

Chapter 7 (joint work with Pietro Perona) takes a simpler but more general model than that presented in Chapter 3 and applies it to binary, discrete multi-valued, and bounding box annotations. It then shows how the model can be used to actively query annotators for labels until a predetermined level of confidence has been reached. By learning the competences of the annotators in the crowd, it shows how to pick out the best performers and obtain high quality annotations with very few labels.

Chapter 8 (joint work with Pietro Perona) explores the problem of estimating machine performance using as few expert-provided labels as possible. Specifically, the confidence scores produced by binary classifiers are modeled by a Bayesian mixture model. By taking a semisupervised approach, it possible to obtain accurate estimates of classifier performance using as few as 10 labels.

Chapter 2

Image Annotation

To turn an image into structured information, the objects in the image and the properties of the image as a whole must to be annotated. Annotation may mean a lot of things, but in this work we focus on two main issues: (1) categorizing and measuring properties of images and (2) categorizing and measuring properties of objects in the images. In this chapter, we describe various annotation types, what user interfaces are needed to produce them, and how to train annotators to provide annotations accurately.

2.1 Annotation Types

Annotations can be applied at different levels of resolution. For many applications, it is useful to know whether an object of interest is present in an image, or whether the image belongs to a certain category. In those cases, the annotation is for the whole image, and so is an "image-level" annotation. For other tasks, it is necessary to know properties of an object, such as its location, size, or configuration. We say that such annotations are at the "object-level." Here we list different types of annotations, and provide examples of their use.

Binary: Binary annotation is defined as asking a binary question about an image or an object. For example, "Does the image belong to category X?" or "Is object category Y present in the image?" It is one of the most widely used annotation types.

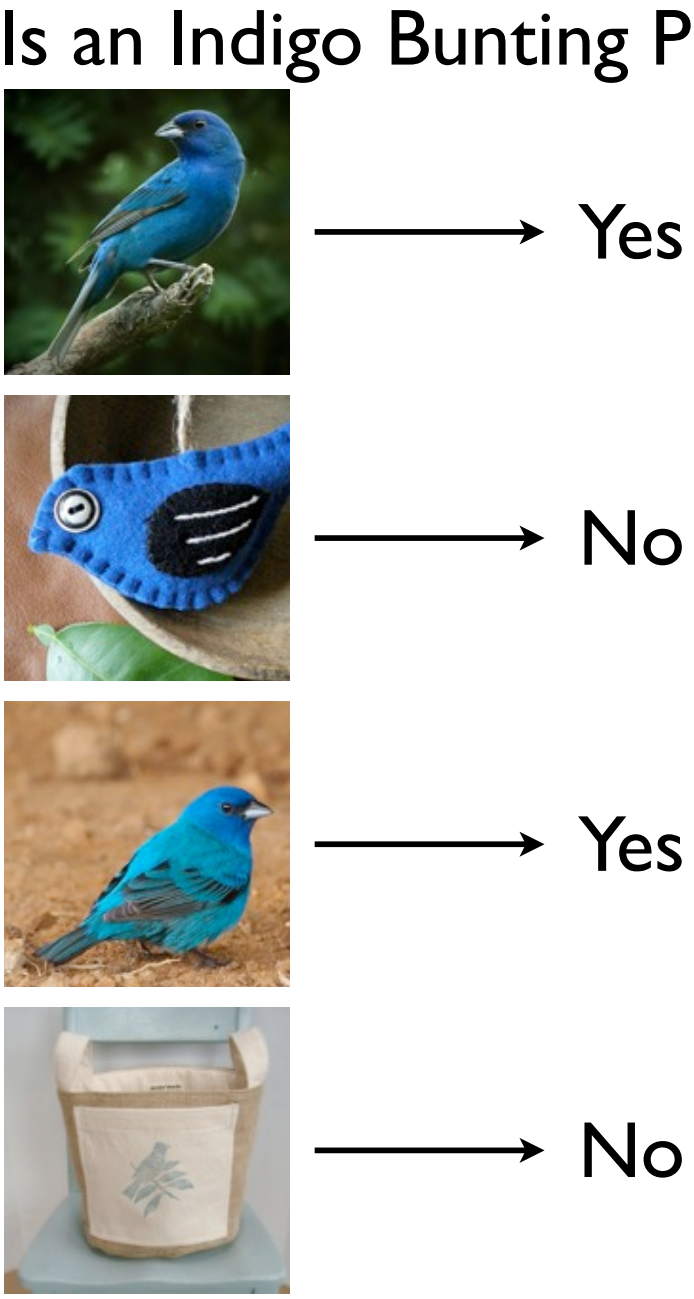

Figure 2.1: Example of a binary filtering task. Flickr.com was queried for the bird species "Indigo Bunting." Four resulting images are shown. For each image, the binary question "Is an Indigo Bunting present?" is asked. The correct answers are shown in the right column.

It is often used for "filtering" multi-category datasets. For example, the CUB-200 dataset [WBM+10] is a dataset with example images for 200 bird species. In order to create it, images were found on Flickr.com[1] by querying for images with tags matching the species name. However, since not all images returned by Flickr.com were actually of the species of interest, a binary question had to be asked for each image: "Is there a bird of species X in the image?" (see Figure 2.1). Another example of binary labels in the literature is the Animals with Attributes dataset [LNH09], where 50 animal classes were labeled with 85 binary attributes. Similarly, the CUB-200 dataset also has binary attribute labels for each image.

***n*-ary:** By extending the number of answer choices from two to many, we have "*n*-ary" annotations (also known as discrete, multiple choice or polytomous labels). For example, the choice may be to assign an image to one out of 101 object categories, as for the Caltech 101 dataset [FFFP04]. If the choices are ordered, as is the case for confidence labels, they are known as "ordinal categorical labels." For example, one popular confidence scale is the Likert-scale [Lik32] where the choices are: Strongly Agree, Agree, Neither Agree or Disagree, Disagree, Strongly Disagree.

Location: The location of an object, usually defined as the geometric center of the extent of the object, can be annotated by placing a "marker" on the object (see Figure 2.2). The marker could take the shape of a cross, box or circle, and is placed on the image by clicking on it.

Extent: Since photographs of the real world have perspective, and so objects can vary in size, scale is usually required in addition to location. While scale can be parameterized by a single scalar, a more popular type of annotation is the "bounding box," which parameterized the vertical and horizontal extent of the object in the image (see Figure 2.2). In computer vision, datasets with bounding box annotations are very common [EVGW+10, DWSP11, DDS+09].

[1] `http://www.flickr.com/`

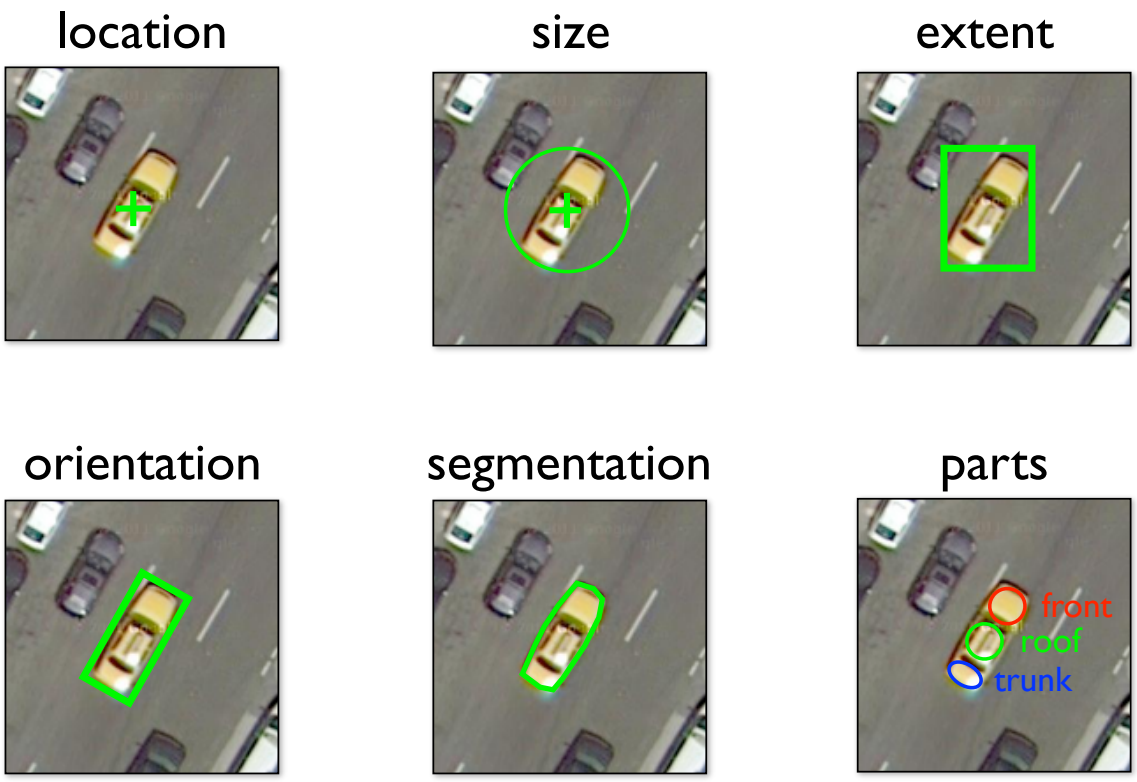

Figure 2.2: Object-level annotation types. Starting from the upper left example: location can be indicated by placing a marker in the middle of the object. Size is indicated by fitting the object within a circle. The extent of the object can be annotated by fitting a "bounding box" snugly around the object. To indicate the orientation, the bounding box can be rotated. Segmentation annotations are used to denote which pixels belong to the object. Finally, part annotations are useful for describing the configuration of the object.

Orientation: In scientific applications, it may be necessary to know the orientation of an object. For example, if the task is to characterize animal social behavior, it it useful to know in which direction an animal is looking [BRB+09, DWP10]. The orientation can be indicated by rotating a box or an ellipse to align with the object (see Figure 2.2).

Segmentation: For a finer description of an objects extent, it is possible to "segment" the object from the background (see Figure 2.2). Polygons or contours can be used to denote the boundary of the object. For pixel-level detail, an alternative representation is to use a binary mask with the same dimensions as the image [FFFP04].

Parts: Since many objects can be thought of as made up of parts [FPZ03], there exists datasets with rich part annotations [BM09]. Most often, the parts are represented by location only (see Figure 2.2). In some cases, for convenience, the parts are represented as a connected skeleton [BM09].

Grouping: Grouping annotations are useful for denoting which images are similar to each other [GWKP11a, TLB+11]. Implicitly, grouping annotations can be thought of as providing pairwise annotations between all the images that are being grouped (either images are in the same group, or they are not).

2.2 Annotation User Interfaces

When outsourcing annotation tasks, it is important to design fast and intuitive graphical user interfaces (GUIs). The reason is that annotators on services like MTurk, like most people, value their time highly. Slow or complicated interfaces will either scare away good annotators, or result in lower throughput at the same price. In this section we describe some of the lessons we have learned through our work with building web-based interfaces for image annotation, and from surveying GUIs we have encountered in our research. Developing a good GUI is more of an art than a science. In many

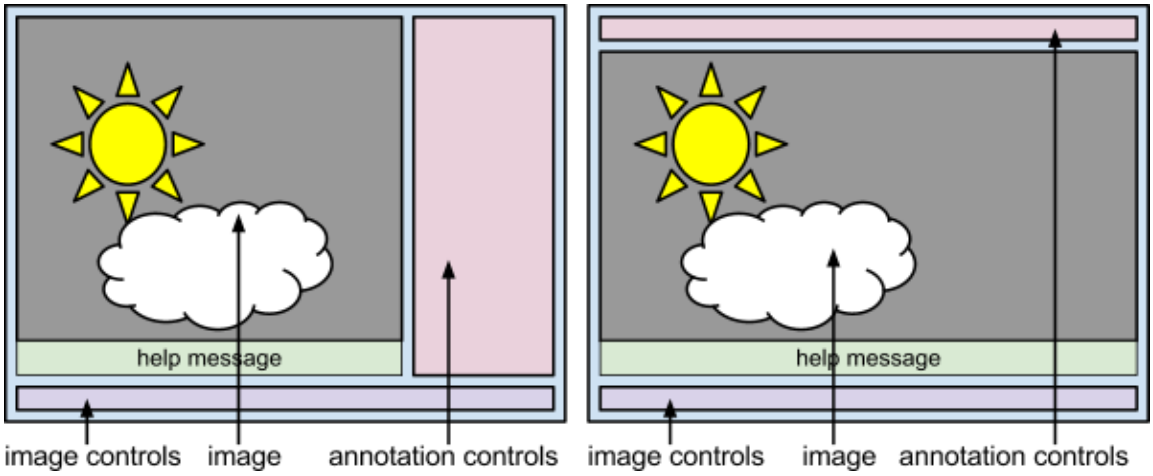

Figure 2.3: Examples of annotation GUI layouts. There are three main components in the annotation GUI: (1) The image to be annotated. (2) Image controls, including buttons for changing zoom, contrast and color channel. (3) Annotation controls. If the annotation is at the image-level, the annotation control may include binary or n-ary questions. If the annotation is at the object-level, the annotation controls may include buttons for changing the object category or for selecting or removing markers from the image. Which layout to choose depends on how many annotation controls are needed.

cases, design choices have to be made not based on studies or data, but based on intuition and user feedback; there is simply not enough time to justify every choice. Thus, while our findings might have general implications, they may not directly apply to other annotation tasks.

An annotation GUI consists of three main components (see Figure 2.3): (1) the image to be annotated, (2) image controls, and (3) annotation controls. In most cases, the image is zoomable and the user can pan the image. Zoom level, image contrast, and choice of color channels can be changed using buttons in the image control component. Selecting, deleting and hiding annotation markers can be done using the marker control.

Usually, the annotator is asked to annotate many images in a series. In that case, the annotation GUI must be placed inside a "navigation GUI" (see Figure 2.4). The navigation GUI lets the user navigate between the images to be annotated and access instructions and example annotations .

Figure 2.4: Navigating an annotation task using a navigation GUI. Top: Mockup of a typical annotation GUI with components for navigating the images, some brief instructions, buttons to access example annotations and comprehensive instructions, and the annotation GUI (see Figure 2.3). Most of the components are optional, but may aid the annotator depending on the task. Bottom: Real instantiation of an annotation GUI with a slightly different layout.

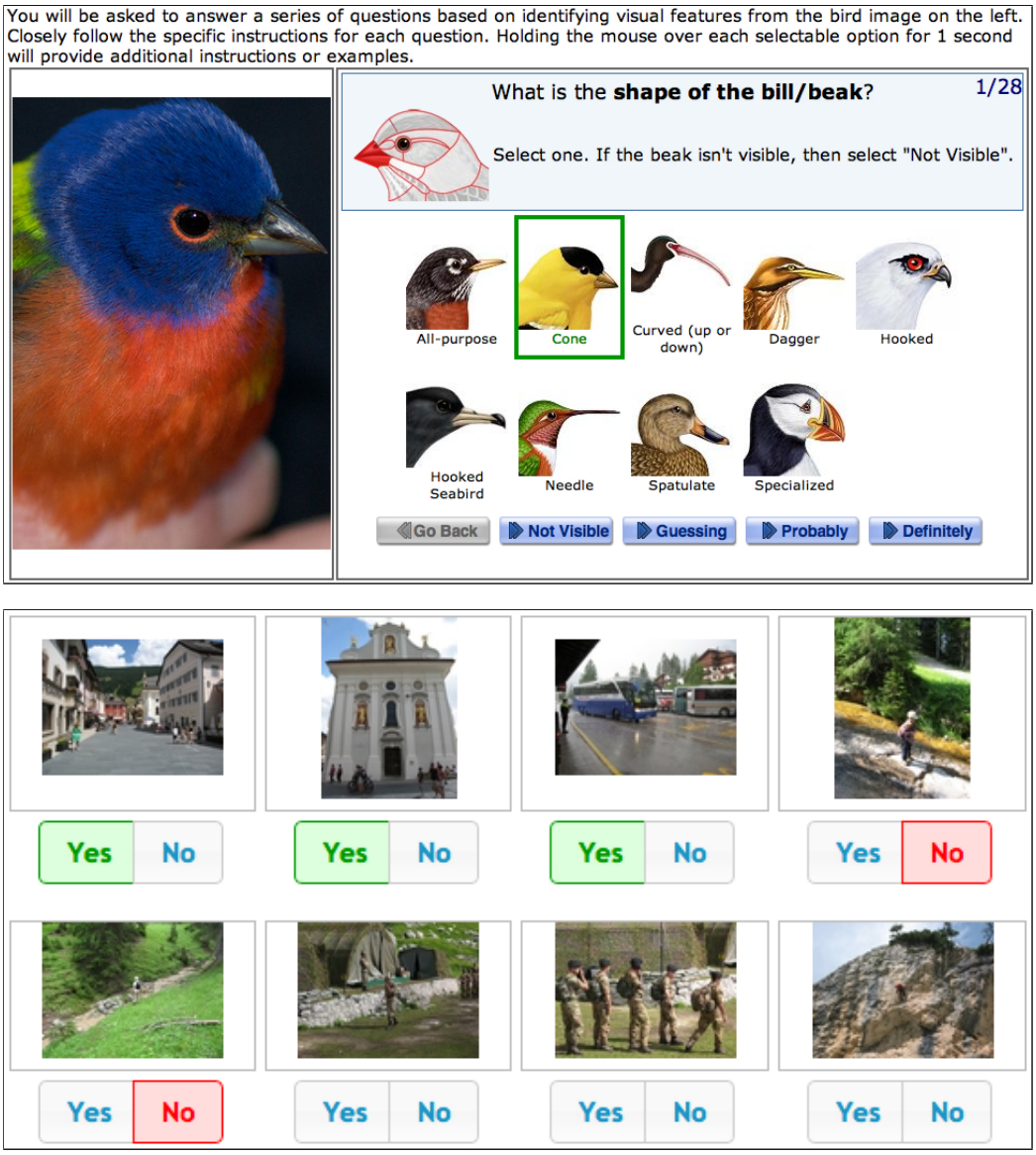

Figure 2.5: Examples of image-level annotation GUIs. Top: For n-ary questions, it may be difficult to fit more than one image at a time on the computer screen. Bottom: For simple binary questions, it is sometimes possible to make the annotation process faster by showing multiple smaller images in a single view. In this example, annotators were asked to select images containing houses.

A GUI for providing image-level annotations is quite simple. The GUI is only required to show two things: the image to be annotated and the questions asked about the image (see Figure 2.5-top). If only a single question is asked per image, the annotation process can be sped up by fitting multiple images in a single view (see Figure 2.5-bottom).

Providing object-level annotations is usually done by placing "markers" on the image. For location annotations, it is enough to mouse-click on the image to place a marker indicating an object location. However, if the marker also indicates scale or extent, more complex mouse interactions are needed. Ideally, the mouse interaction should involve as few mouse clicks as possible, because each click has a time-cost. We have found that a mouse press-drag-release interaction sequence is appropriate for most annotation types: the user presses down the mouse button when the mouse pointer is somewhere on the image where he wishes to locate the marker, as he drags the mouse, the extent of the marker is changed. Upon release of the mouse button, the location and extent of the marker is fixed. Depending on what kind of marker primitive is being annotated, the same interaction produces a different behavior (see Figure 2.6). In some cases, depending on the task, the same marker primitive may have a different interaction behavior (see Figure 2.7).

Object-level annotations may also include discrete or textual labels. In that case, the labels can be provided by clicking on the marker annotation. The mouse-click brings up a menu showing category labels or a text box for changing a textual label (see Figure 2.8).

2.3 Annotator Training and Testing

In order to obtain high quality annotations from a crowd of non-expert annotators, it is essential that they are trained well for the task. In our experience, the result is otherwise that they use the annotation GUI incorrectly, or look for the wrong things in the images. This section describes some design patterns that we have found useful for training annotators in new tasks and making sure that they perform well.

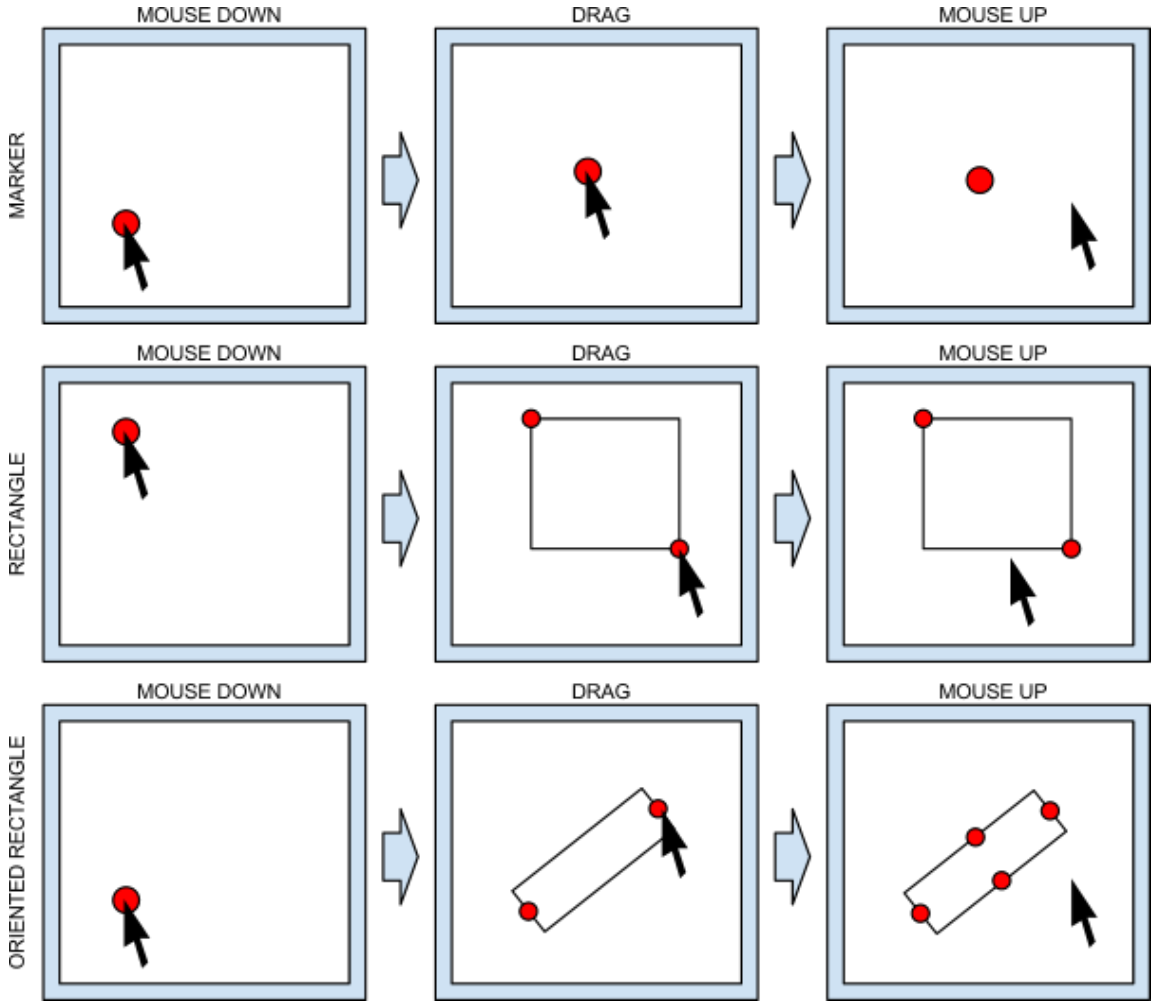

Figure 2.6: Mouse interactions for adding new markers to the image. Each row shows a different marker primitive: simple markers, rectangles and oriented rectangles. The columns show the three steps of the user pressing and holding down the mouse button, dragging the pointer, and releasing the button. The interaction is different for each marker primitive.

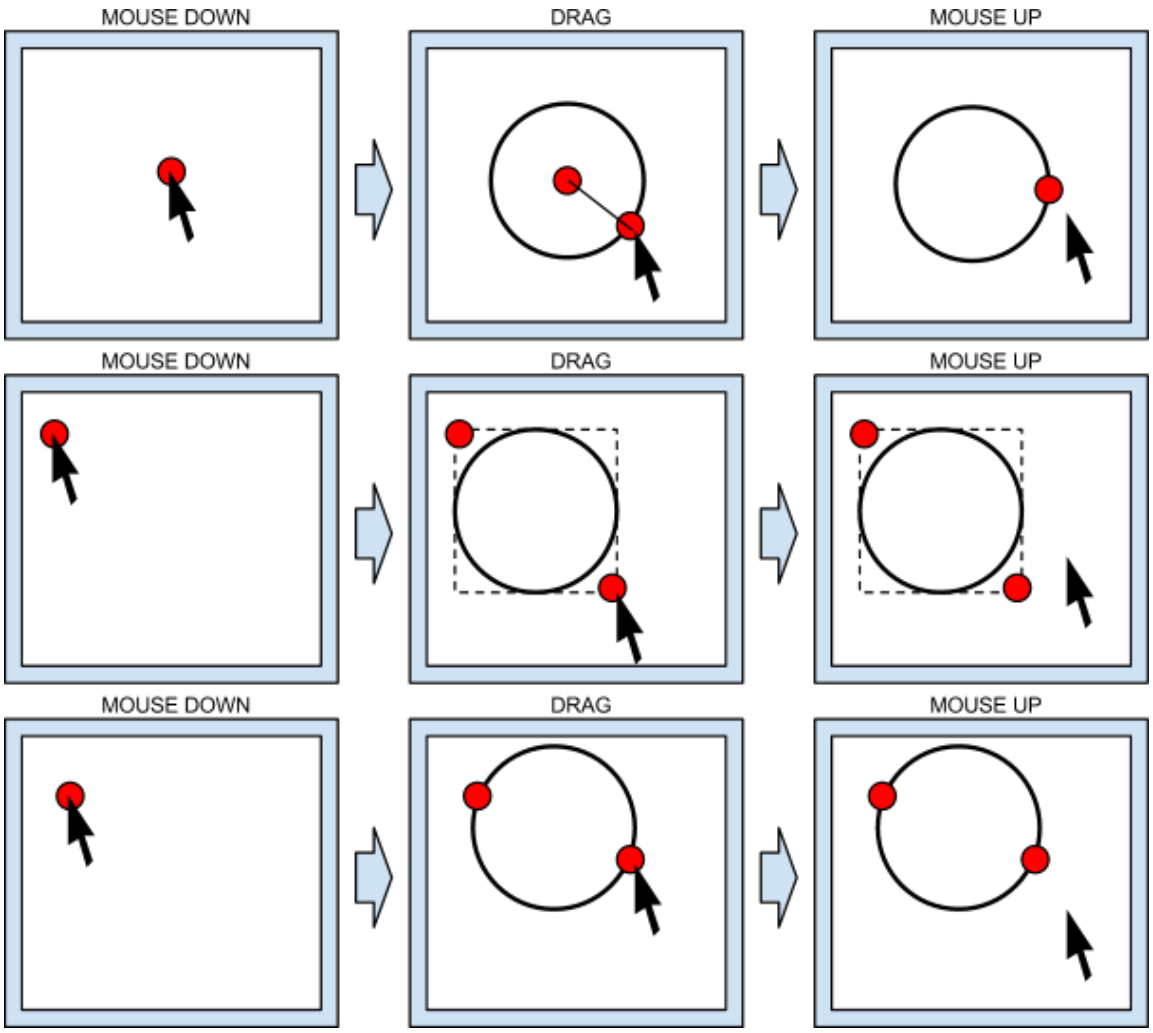

Figure 2.7: Same marker primitive (ellipse), but different mouse interaction behavior. The behavior described pictorially in the top row may be most appropriate for placing fixed-radius ellipses. The middle row is most appropriate in situations where the orientation of the ellipse is always fixed. The behavior in the bottom row works well when the orientation of the ellipse is important.

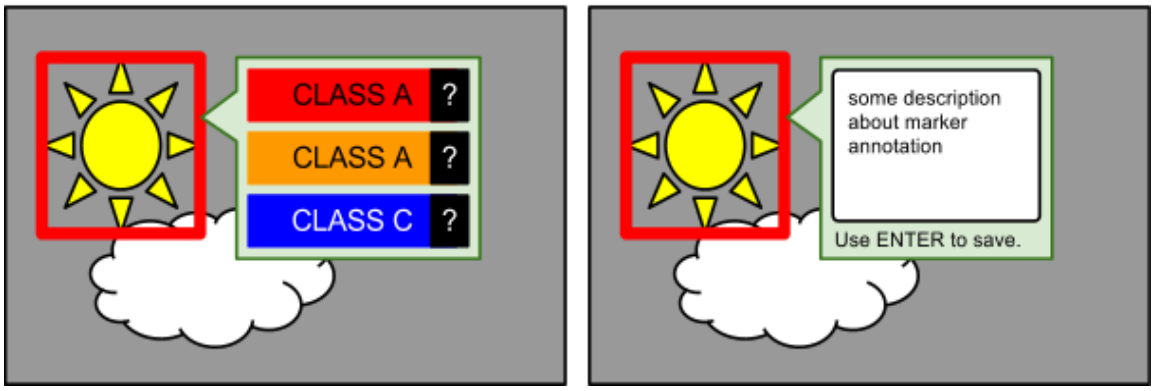

Figure 2.8: Adding object-level discrete and textual annotations. By clicking on annotation, the user is shown a menu to add or modify object-level annotations, such as category labels (left) and text labels (right).

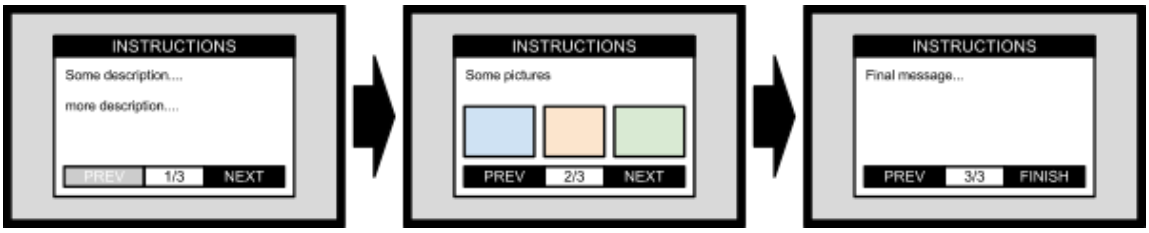

Figure 2.9: Instructions split into multiple slides in the instruction-step of the training session.

In the training session, the annotators need clear instructions of what is required from them. Part of this can be achieved by written instructions, but usually annotated example images are even more effective. Video instructions showing examples of how to use the annotation GUI are also very well received by annotators. However, to really learn how to annotate correctly, one of the best approaches is to have the annotators try out the task for themselves and get feedback on their performance. Here we suggest splitting the training session into three steps:

1. **Instruction-step:** In the instruction-step, the annotator reads written instructions or watches an instruction video. If using a service like MTurk, it is important to indicate how long the annotator is expected to work on the task (so that he can calculate the hourly rate), and how the work will be judged (since MTurk allows requesters to reject and not pay for poor work). Since annotators

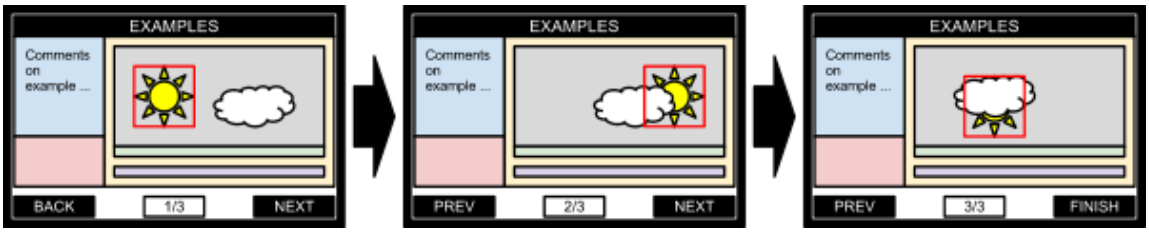

Figure 2.10: In the examples-step of the training session, the annotator studies annotated example images.

on MTurk are often rushed, textual instructions should be clear and concise. Splitting the instructions into multiple slides in a slide show and writing instructions as brief bullet points is very effective (see Figure 2.9).

2. **Example-step:** The example-step shows multiple annotated example images (see Figure 2.10). It lets the annotator study in more detail how to annotate the images correctly.

3. **Trial-step:** The purpose of the trial-step is to give the annotator feedback on how well he understands the task and the GUI. One of the simplest ways to accomplish this is to let the annotator annotate an image according to the instructions, and then show his own and the correct answer (provided by an expert) side-by-side (see Figure 2.11). A more complicated scheme is to write software to catch errors, and provide textual feedback on the mistakes the annotator is making (see Figure 2.12).

At any point in the training session, the annotator can go back to view the instructions or the examples (see Figure 2.13). Once the training has been completed, the annotator is taken to annotate the real images.

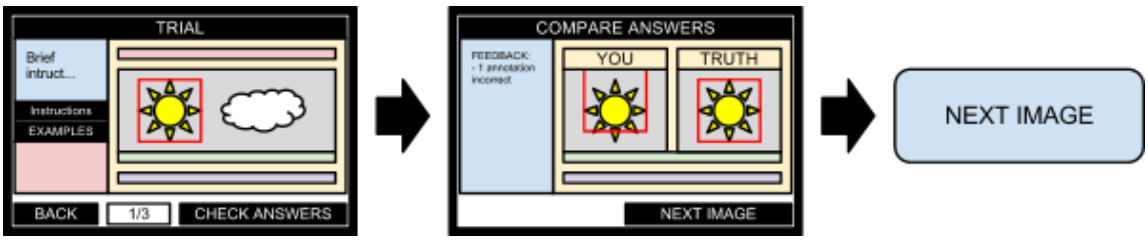

Figure 2.11: Simple instantiation of the trial-step. The annotator makes an attempt at annotating the image according to the instructions. He then clicks on the "check answers" button and is shown his own and the correct answer (as provided by an expert) side-by-side. This lets the annotator compare what mistakes he made.

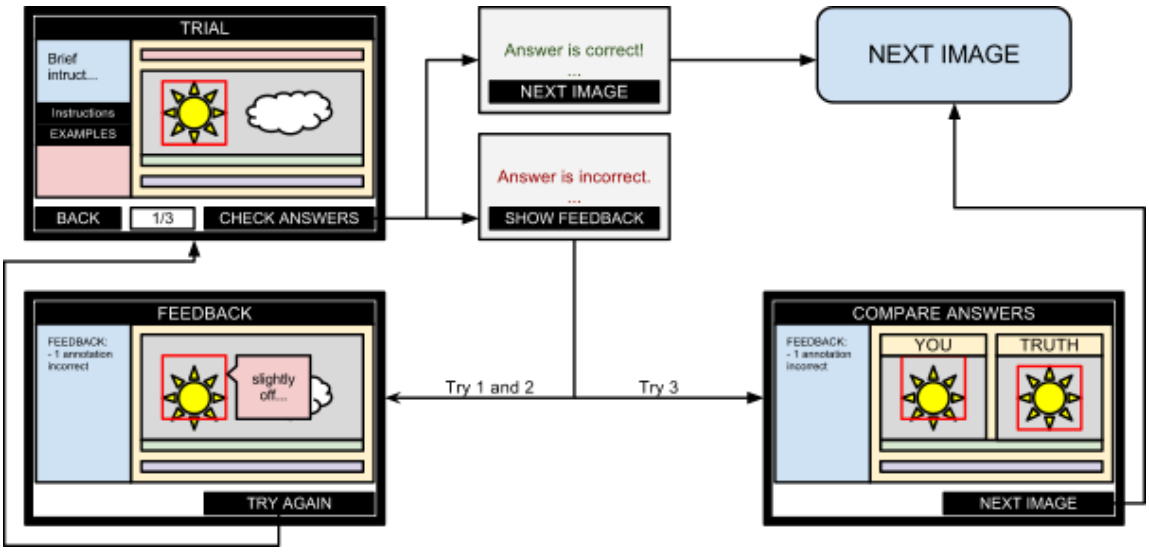

Figure 2.12: Trial-step with detailed feedback. The annotator makes an attempt at annotating the image according to the instructions and then clicks on the "check answers" button. If his answers are correct, he is taken to the next image. If not, he is shown feedback on the annotations he has made, and is encouraged to try again. After failing a few times, the annotator is taken to the interface in Figure 2.11.

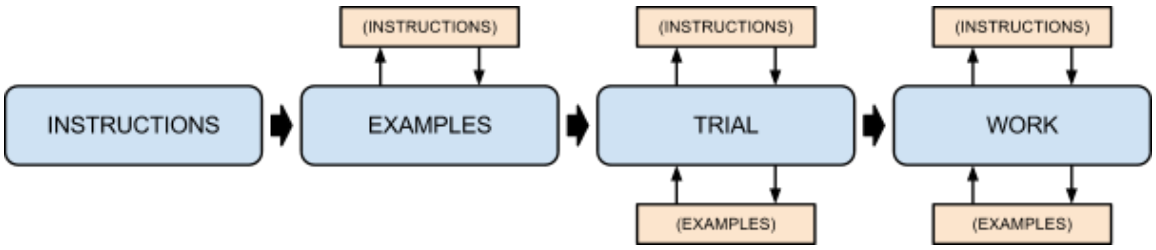

Figure 2.13: Training session work flow. The annotator starts by reading the instructions, and is then shown some example annotations. In the last step, the trial-step, the annotator is challenged to try out a few images and get feedback on how well he understood the task. Once the training session is concluded, the annotator is taken to work on the real image. At any point the annotator can go back to view the instructions or example annotations.

Chapter 3

Binary Annotations

3.1 Abstract

Distributing labeling tasks among hundreds or thousands of annotators is an increasingly important method for annotating large datasets. We present a method for estimating the underlying value (e.g., the class) of each image from (noisy) annotations provided by multiple annotators. Our method is based on a model of the image formation and annotation process. Each image has different characteristics that are represented in an abstract Euclidean space. Each annotator is modeled as a multi-dimensional entity with variables representing competence, expertise and bias. This allows the model to discover and represent groups of annotators that have different sets of skills and knowledge, as well as groups of images that differ qualitatively. We find that our model predicts ground truth labels on both synthetic and real data more accurately than state of the art methods. Experiments also show that our lmodel, starting from a set of binary labels, may discover rich information, such as different "schools of thought" amongst the annotators, and can group together images belonging to separate categories.

3.2 Introduction

Producing large-scale training, validation and test sets is vital for many applications. Most often this job has to be carried out "by hand" and thus it is delicate, expensive,

and tedious. Services such as Amazon Mechanical Turk (MTurk) have made it easy to distribute simple labeling tasks to hundreds of workers. Such "crowdsourcing" is increasingly popular and has been used to annotate large datasets in, for example, Computer Vision [SF08] and Natural Language Processing [SOJN08]. As some annotators are unreliable, the common wisdom is to collect multiple labels per exemplar and rely on "majority voting" to determine the correct label. We propose a model for the annotation process with the goal of obtaining more reliable labels with as few annotators as possible.

It has been observed that some annotators are more skilled and consistent in their labels than others. We postulate that the ability of annotators is multidimensional; that is, an annotator may be good at some aspects of a task but worse at others. Annotators may also attach different costs to different kinds of errors, resulting in different biases for the annotations. Furthermore, different pieces of data may be easier or more difficult to label. All of these factors contribute to a "noisy" annotation process resulting in inconsistent labels. Although approaches for modeling certain aspects of the annotation process have been proposed in the past [DS79, SPI08, SFB+95, SP08, WRW+09, RYZ+09, WP10], no attempt has been made to blend all characteristics of the process into a single unified model.

This paper has two main contributions: (1) we improve on current state-of-the-art methods for crowdsourcing by introducing a more comprehensive and accurate model of the human annotation process, and (2) we provide insight into the human annotation process by learning a richer representation that distinguishes amongst the different sources of annotator error. Understanding the annotation process can be important toward quantifying the extent to which datasets constructed from human data are "ground truth".

We propose a generative Bayesian model for the annotation process. We describe an inference algorithm to estimate the properties of the data being labeled and the annotators labeling them. We show on synthetic and real data that the model can be used to estimate data difficulty and annotator biases, while identifying annotators' different "areas of strength". While many of our results are valid for general labels

and tasks, we focus on the binary labeling of images.

3.3 Related Work

The advantages and drawbacks of using crowdsourcing services for labeling large datasets have been explored by various authors [DDS+09, SOJN08, SF08]. In general, it has been found that many labels are of high quality [SF08], but a few sloppy annotators do low quality work [SOJN08, WP10]; thus the need for efficient algorithms for integrating the labels from many annotators [SPI08, WP10]. A related topic is that of using paired games for obtaining annotations, which can be seen as a form of crowdsourcing [vAD04, vAMM+08].

Methods for combining the labels from many different annotators have been studied before. Dawid and Skene [DS79] presented a model for multi-valued annotations where the biases and skills of the annotators were modeled by a confusion matrix. This model was generalized and extended to other annotation types by Welinder and Perona [WP10]. Similarly, the model presented by Raykar et al. [RYZ+09] considered annotator bias in the context of training binary classifiers with noisy labels. Building on these works, our model goes a step further in modeling each annotator as a multidimensional classifier in an abstract feature space. We also draw inspiration from Whitehill et al. [WRW+09], who modeled both annotator competence and image difficulty, but did not consider annotator bias. Our model generalizes [WRW+09] by introducing a high-dimensional concept of image difficulty and combining it with a broader definition of annotator competence. Other approaches have been proposed for non-binary annotations [SP08, SFB+95, WP10]. By modeling annotator competence and image difficulty as multidimensional quantities, our approach achieves better performance on real data than previous methods and provides a richer output space for separating groups of annotators and images.

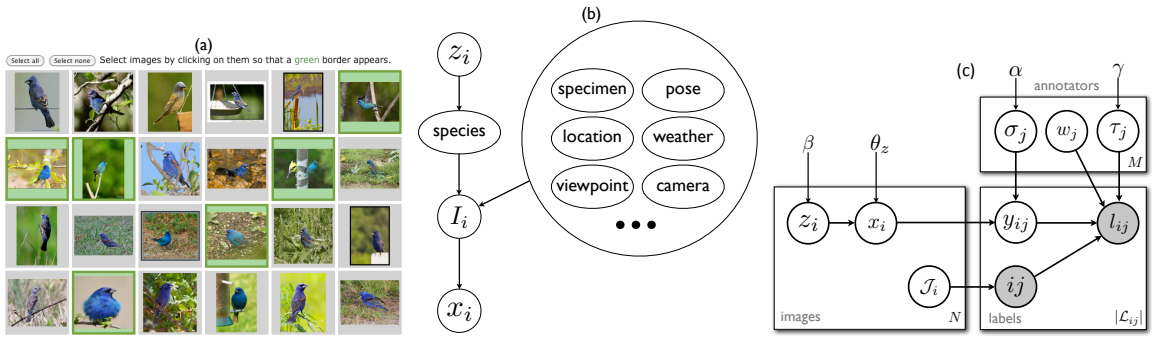

Figure 3.1: (a) Sample MTurk task where annotators were asked to click on images of Indigo Bunting (described in Section 3.6.2). (b) The image formation process. The class variable z_i models if the object (Indigo Bunting) will be present ($z_i = 1$) or absent ($z_i = 0$) in the image, while a number of "nuisance factors" influence the appearance of the image. The image is then transformed into a low-dimensional representation x_i which captures the main attributes that are considered by annotators in labeling the image. (c) Probabilistic graphical model of the entire annotation process where image formation is summarized by the nodes z_i and x_i. The observed variables, indicated by shaded circles, are the index i of the image, index j of the annotators, and value l_{ij} of the label provided by annotator j for image i. The annotation process is repeated for all i and for multiple j thus obtaining multiple labels per image with each annotator labeling multiple images (see Section 3.4).

3.4 The Annotation Process

An annotator, indexed by j, looks at image I_i and assigns it a label l_{ij}. *Competent* annotators provide accurate and precise labels, while unskilled annotators provide inconsistent labels. There is also the possibility of adversarial annotators assigning labels that are opposite to those assigned by competent annotators. Annotators may have different areas of strength, or *expertise*, and thus provide more reliable labels on different subsets of images. For example, when asked to label images containing ducks some annotators may be more aware of the distinction between ducks and geese while others may be more aware of the distinction between ducks, grebes, and cormorants (visually similar bird species). Furthermore, different annotators may weigh errors differently; one annotator may be intolerant of false positives, while another is more *optimistic* and accepts the cost of a few false positives in order to get a higher detection rate. Lastly, the *difficulty* of the image may also matter. A difficult or ambiguous image may be labeled inconsistently even by competent annotators, while an easy image is labeled consistently even by sloppy annotators. In modeling the annotation process, all of these factors should be considered.

We model the annotation process in a sequence of steps. N images are produced by some image capture/collection process. First, a variable z_i decides which set of "objects" contribute to producing an image I_i. For example, $z_i \in \{0, 1\}$ may denote the presence/absence of a particular bird species. A number of "nuisance factors," such as viewpoint and pose, determine the image (see Figure 3.1).

Each image is transformed by a deterministic "visual transformation" converting pixels into a vector of task-specific measurements x_i, representing measurements that are available to the visual system of an ideal annotator. For example, the x_i could be the firing rates of task-relevant neurons in the brain of the best human annotator. Another way to think about x_i is that it is a vector of visual attributes (beak shape, plumage color, tail length, etc.) that the annotator will consider when deciding on a label. The process of transforming z_i to the "signal" x_i is stochastic and it is parameterized by θ_z, which accounts for the variability in image formation due to the

nuisance factors.

There are M annotators in total, and the set of annotators that label image i is denoted by $\mathcal{J}_i$. An annotator $j \in \mathcal{J}_i$, selected to label image I_i, does not have direct access to x_i, but rather to $y_{ij} = x_i + n_{ij}$, a version of the signal corrupted by annotator-specific and image-specific "noise" n_{ij}. The noise process models differences between the measurements that are ultimately available to individual annotators. These differences may be due to visual acuity, attention, direction of gaze, etc. The statistics of this noise are different from annotator to annotator and are parametrized by σ_j. Most significantly, the variance of the noise will be lower for competent annotators, as they are more likely to have access to a clearer and more consistent representation of the image than confused or unskilled annotators.

The vector y_{ij} can be understood as a perceptual encoding that encompasses all major components that affect an annotator's judgment on an annotation task. Each annotator is parameterized by a unit vector $\hat{w}_j$, which models the annotator's individual weighting on each of these components. In this way, $\hat{w}_j$ encodes the training or expertise of the annotator in a multidimensional space. The scalar projection $\langle y_{ij}, \hat{w}_j \rangle$ is compared to a threshold $\hat{\tau}_j$. If the signal is above the threshold, the annotator assigns a label $l_{ij} = 1$, and $l_{ij} = 0$ otherwise.

3.5 Model and Inference

Putting together the assumptions of the previous section, we obtain the graphical model shown in Figure 3.1. We will assume a Bayesian treatment, with priors on all parameters. The joint probability distribution, excluding hyper-parameters for brevity, can be written as

$$p(\mathcal{L}, z, x, y, \sigma, \hat{w}, \hat{\tau}) = \prod_{j=1}^{M} p(\sigma_j) p(\hat{\tau}_j) p(\hat{w}_j) \prod_{i=1}^{N} \left(p(z_i) p(x_i \mid z_i) \prod_{j \in \mathcal{J}_i} p(y_{ij} \mid x_i, \sigma_j)\, p(l_{ij} \mid \hat{w}_j, \hat{\tau}_j, y_{ij}) \right), \tag{3.1}$$

where we denote z, x, y, σ, $\hat{\tau}$, $\hat{w}$, and $\mathcal{L}$ to mean the sets of all the corresponding subscripted variables. This section describes further assumptions on the probability

distributions. These assumptions are not necessary; however, in practice they simplify inference without compromising the quality of the parameter estimates.

Although both z_i and l_{ij} may be continuous or multivalued discrete in a more general treatment of the model [WP10], we henceforth assume that they are binary, i.e., $z_i, l_{ij} \in \{0, 1\}$. We assume a Bernoulli prior on z_i with $p(z_i = 1) = \beta$, and that x_i is normally distributed[1] with variance θ_z^2,

$$p(x_i \mid z_i) = \mathcal{N}(x_i;\, \mu_z, \theta_z^2), \tag{3.2}$$

where $\mu_z = -1$ if $z_i = 0$ and $\mu_z = 1$ if $z_i = 1$ (see Figure 3.2a). If x_i and y_{ij} are multi-dimensional, then σ_j is a covariance matrix. These assumptions are equivalent to using a mixture of Gaussians prior on x_i.

The noisy version of the signal x_i that annotator j sees, denoted by y_{ij}, is assumed to be generated by a Gaussian with variance σ_j^2 centered at x_i, that is, $p(y_{ij} \mid x_i, \sigma_j) = \mathcal{N}(y_{ij};\, x_i, \sigma_j^2)$ (see Figure 3.2b). We assume that each annotator assigns the label l_{ij} according to a linear classifier. The classifier is parameterized by a direction $\hat{w}_j$ of a decision plane and a bias $\hat{\tau}_j$. The label l_{ij} is deterministically chosen, i.e., $l_{ij} = \mathbb{I}\left(\langle \hat{w}_j, y_{ij} \rangle \geq \hat{\tau}_j\right)$, where $\mathbb{I}\left(\cdot\right)$ is the indicator function. It is possible to integrate out y_{ij} and put l_{ij} in direct dependence on x_i,

$$p(l_{ij} = 1 \mid x_i, \sigma_j, \hat{\tau}_j) = \Phi\left(\frac{\langle \hat{w}_j, x_i \rangle - \hat{\tau}_j}{\sigma_j}\right), \tag{3.3}$$

where $\Phi(\cdot)$ is the cumulative standardized normal distribution, a sigmoidal-shaped function.

In order to remove the constraint on $\hat{w}_j$ being a direction, i.e., $\|\hat{w}_j\|^2 = 1$, we reparameterize the problem with $w_j = \hat{w}_j / \sigma_j$ and $\tau_j = \hat{\tau}_j / \sigma_j$. Furthermore, to regularize w_j and τ_j during inference, we give them Gaussian priors parameterized by α and γ respectively. The prior on τ_j is centered at the origin and is very broad ($\gamma = 3$). For the prior on w_j, we kept the center close to the origin to be initially pessimistic of

[1] We used the parameters $\beta = 0.5$ and $\theta_z = 0.8$.

the annotator competence, and to allow for adversarial annotators (mean 1, std 3). All of the hyperparameters were chosen somewhat arbitrarily to define a scale for the parameter space, and in our experiments we found that results (such as error rates in Figure 3.3) were quite insensitive to variations in the hyperparameters. The modified Equation 3.1 becomes,

$$p(\mathcal{L}, x, w, \tau) = \prod_{j=1}^{M} p(\tau_j \mid \gamma) p(w_j \mid \alpha) \prod_{i=1}^{N} \left(p(x_i \mid \theta_z, \beta) \prod_{j \in \mathcal{J}_i} p(l_{ij} \mid x_i, w_j, \tau_j) \right). \quad (3.4)$$

The only observed variables in the model are the labels $\mathcal{L} = \{l_{ij}\}$, from which the other parameters have to be inferred. Since we have priors on the parameters, we proceed by MAP estimation, where we find the optimal parameters $(x^\star, w^\star, \tau^\star)$ by maximizing the posterior on the parameters,

$$(x^\star, w^\star, \tau^\star) = \arg\max_{x,w,\tau} p(x, w, \tau \mid \mathcal{L}) = \arg\max_{x,w,\tau} m(x, w, \tau), \quad (3.5)$$

where we have defined $m(x, w, \tau) = \log p(\mathcal{L}, x, w, \tau)$ from Equation 3.4. Thus, to do inference, we need to optimize

$$\begin{aligned} m(x, w, \tau) = &\sum_{i=1}^{N} \log p(x_i \mid \theta_z, \beta) + \sum_{j=1}^{M} \log p(w_j \mid \alpha) + \sum_{j=1}^{M} \log p(\tau_j \mid \gamma) \\ &+ \sum_{i=1}^{N} \sum_{j \in \mathcal{J}_i} \left[l_{ij} \log \Phi\left(\langle w_j, x_i \rangle - \tau_j\right) + (1 - l_{ij}) \log\left(1 - \Phi\left(\langle w_j, x_i \rangle - \tau_j\right)\right) \right]. \end{aligned} \quad (3.6)$$

To maximize (3.6) we carry out alternating optimization using gradient ascent. We begin by fixing the x parameters and optimizing Equation 3.6 for (w, τ) using gradient ascent. Then we fix (w, τ) and optimize for x using gradient ascent, iterating between fixing the image parameters and annotator parameters back and forth. Empirically, we have observed that this optimization scheme usually converges within 20 iterations.

In the derivation of the model above, there is no restriction on the dimensionality of x_i and w_j; they may be one-dimensional scalars or higher-dimensional vectors.

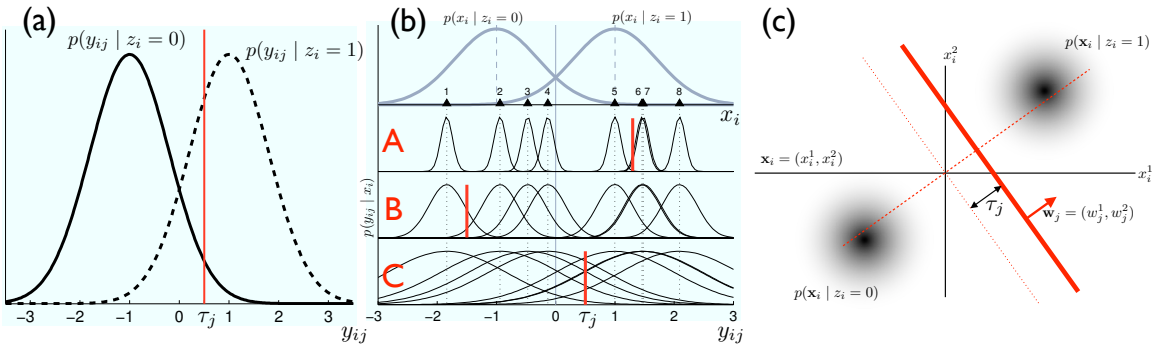

Figure 3.2: Assumptions of the model. (a) Labeling is modeled in a signal detection theory framework, where the signal y_{ij} that annotator j sees for image I_i is produced by one of two Gaussian distributions. Depending on y_{ij} and annotator parameters w_j and τ_j, the annotator labels 1 or 0. (b) The image representation x_i is assumed to be generated by a Gaussian mixture model where z_i selects the component. The figure shows 8 different realizations x_i $(x_1, \ldots, x_8)$, generated from the mixture model. Depending on the annotator j, noise n_{ij} is added to x_i. The three lower plots shows the noise distributions for three different annotators (A,B,C), with increasing "incompetence" σ_j. The biases τ_j of the annotators are shown with the red bars. Image no. 4, represented by x_4, is the most ambiguous image, as it is very close to the optimal decision plane at $x_i = 0$. (c) An example of 2-dimensional x_i. The red line shows the decision plane for one annotator.

In the former case, assuming $\hat{w}_j = 1$, the model is equivalent to a standard signal detection theoretic model [GS66] where a signal y_{ij} is generated by one of two Normal distributions $p(y_{ij} \mid z_i) = \mathcal{N}(y_{ij} \mid \mu_z, s^2)$ with variance $s^2 = \theta_z^2 + \sigma_j^2$, centered on $\mu_0 = -1$ and $\mu_1 = 1$ for $z_i = 0$ and $z_i = 1$ respectively (see Figure 3.2a). In signal detection theory, the sensitivity index, conventionally denoted d', is a measure of how well the annotator can discriminate the two values of z_i [Wic02]. It is defined as the Mahalanobis distance between μ_0 and μ_1 normalized by s,

$$d' = \frac{\mu_1 - \mu_0}{s} = \frac{2}{\sqrt{\theta_z^2 + \sigma_j^2}}. \tag{3.7}$$

Thus, the lower σ_j, the better the annotator can distinguish between classes of z_i, and the more "competent" he is. The sensitivity index can also be computed directly from the false alarm rate f and hit rate h using $d' = \Phi^{-1}(h) - \Phi^{-1}(f)$ where $\Phi^{-1}(\cdot)$ is the inverse of the cumulative normal distribution [Wic02]. Similarly, the "threshold", which is a measure of annotator bias, can be computed by $\lambda = -\frac{1}{2}\left(\Phi^{-1}(h) + \Phi^{-1}(f)\right)$. A large positive λ means that the annotator attributes a high cost to false positives, while a large negative λ means the annotator avoids false negative mistakes. Under the assumptions of our model, λ is related to τ_j in our model by the relation $\lambda = \hat{\tau}_j / s$.

In the case of higher dimensional x_i and w_j, each component of the x_i vector can be thought of as an attribute or a high level feature. For example, the task may be to label only images with a particular bird species, say "duck", with label 1, and all other images with 0. Some images contain no birds at all, while other images contain birds similar to ducks, such as geese or grebes. Some annotators may be more aware of the distinction between ducks and geese and others may be more aware of the distinction between ducks, grebes and cormorants. In this case, x_i can be considered to be 2-dimensional. One dimension represents image attributes that are useful in the distinction between ducks and geese, and the other dimension models parameters that are useful in distinction between ducks and grebes (see Figure 3.2c). Presumably all annotators see the same attributes, signified by x_i, but they use them differently. The model can distinguish between annotators with preferences for different attributes,

as shown in Section 3.6.2.

Image difficulty is represented in the model by the value of x_i (see Figure 3.2b). If there is a particular ground truth decision plane, (w', τ'), images I_i with x_i close to the plane will be more difficult for annotators to label. This is because the annotators see a noise corrupted version, y_{ij}, of x_i. How well the annotators can label a particular image depends on both the closeness of x_i to the ground truth decision plane and the annotator's "noise" level, σ_j. Of course, if the annotator bias τ_j is far from the ground truth decision plane, the labels for images near the ground truth decision plane will be consistent for that annotator, but not necessarily correct.

3.6 Experiments

3.6.1 Synthetic Data

To explore whether the inference procedure estimates image and annotator parameters accurately, we tested our model on synthetic data generated according to the model's assumptions. Similar to the experimental setup in [WRW+09], we generated 500 synthetic image parameters and simulated between 4 and 20 annotators labeling each image. The procedure was repeated 40 times to reduce the noise in the results.

We generated the annotator parameters by randomly sampling σ_j from a Gamma distribution (shape 1.5 and scale 0.3) and biases τ_j from a Normal distribution centered at 0 with standard deviation 0.5. The direction of the decision plane w_j was $+1$ with probability 0.99 and -1 with probability 0.01. The image parameters x_i were generated by a two-dimensional Gaussian mixture model with two components of standard deviation 0.8 centered at -1 and +1. The image ground truth label z_i, and thus the mixture component from which x_i was generated, was sampled from a Bernoulli distribution with $p(z_i = 1) = 0.5$.

For each trial, we measured the correlation between the ground truth values of each parameter and the values estimated by the model. We averaged Spearman's rank correlation coefficient for each parameter over all trials. The result of the simulated

labeling process is shown Figure 3.3a. As can be seen from the figure, the model estimates the parameters accurately, with the accuracy increasing as the number of annotators labeling each image increases. We repeated a similar experiment with 2-dimensional x_i and w_j (see Figure 3.3b). As one would expect, estimating higher dimensional x_i and w_j requires more data.

We also examined how well our model estimated the binary class values, z_i. For comparison, we also tried three other methods on the same data: a simple majority voting rule for each image, the bias-competence model of [DS79], and the GLAD algorithm from [WRW+09][2], which models 1-d image difficulty and annotator competence, but not bias. As can be seen from Figure 3.3c, our method presents a small but consistent improvement. In a separate experiment (not shown) we generated synthetic annotators with increasing bias parameters τ_j. We found that GLAD performs worse than majority voting when the variance in the bias between different annotators is high ($\gamma \gtrsim 0.8$); this was expected as GLAD does not model annotator bias. Similarly, increasing the proportion of difficult images degrades the performance of the model from [DS79]. The performance of our model points to the benefits of modeling all aspects of the annotation process.

3.6.2 Human Data

We next conducted experiments on annotation results from real MTurk annotators. To compare the performance of the different models on a real discrimination task, we prepared dataset of 200 images of birds (100 with Indigo Bunting, and 100 with Blue Grosbeak), and asked 40 annotators per image if it contained at least one Indigo Bunting; this is a challenging task (see Figure 3.1). The annotators were given a description and example photos of the two bird species. Figure 3.3d shows how the performance varies as the number of annotators per image is increased. We sampled a subset of the annotators for each image. Our model did better than the other approaches also on this dataset.

[2]We used the implementation of GLAD available on the first author's website: http://mplab.ucsd.edu/~jake/ We varied the α prior in their code between 1–10 to achieve best performance.

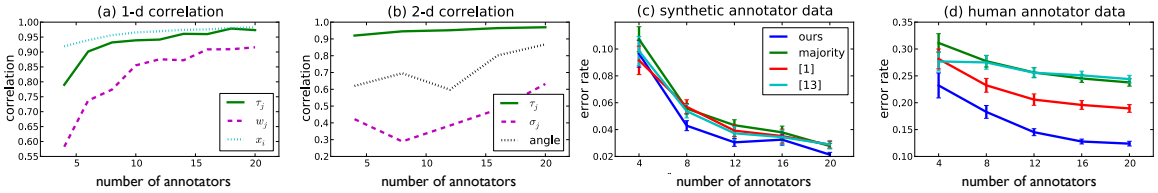

Figure 3.3: (a) and (b) show the correlation between the ground truth and estimated parameters as the number of annotators increases on synthetic data for 1-d and 2-d x_i and w_j. (c) Performance of our model in predicting z_i on the data from (a), compared to majority voting, the model of [DS79], and GLAD [WRW+09]. (d) Performance on real labels collected from MTurk. See Section 3.6.1 for details on (a-c) and Section 3.6.2 for details on (d).

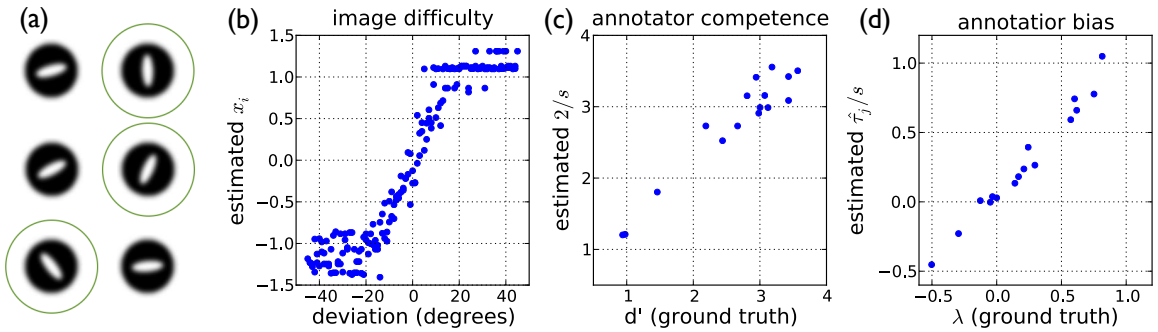

Figure 3.4: Ellipse dataset. (a) The images to be labeled were fuzzy ellipses (oriented uniformly from 0 to π) enclosed in dark circles. The task was to select ellipses that were more vertical than horizontal (the former are marked with green circles in the figure). (b-d) The image difficulty parameters x_i, annotator competence $2/s$, and bias $\hat{\tau}_j/s$ learned by our model are compared to the ground truth equivalents. The closer x_i is to 0, the more ambiguous/difficult the discrimination task, corresponding to ellipses that have close to 45° orientation.

To demonstrate that annotator competence, annotator bias, image difficulty, and multi-dimensional decision surfaces are important real life phenomena affecting the annotation process, and to quantify our model's ability to adapt to each of them, we tested our model on three different image datasets: one based on pictures of rotated ellipses, another based on synthetically generated "greebles", and a third dataset with images of waterbirds.

Ellipse Dataset: Annotators were given the simple task of selecting ellipses which they believed to be more vertical than horizontal. This dataset was chosen to make the model's predictions quantifiable, because ground truth class labels and ellipse angle parameters are known to us for each test image (but hidden from the inference algorithm).

By definition, ellipses at an angle of 45° are impossible to classify, and we expect that images gradually become easier to classify as the angle moves away from 45°. We used a total of 180 ellipse images, with rotation angle varying from 1–180°, and collected labels from 20 MTurk annotators for each image. In this dataset, the estimated image parameters x_i and annotator parameters w_j are 1-dimensional, where the magnitudes encode image difficulty and annotator competence respectively. Since we had ground truth labels, we could compute the false alarm and hit rates for each annotator, and thus compute λ and d' for comparison with $\hat{\tau}_j/s$ and $2/s$ (see Equation 3.7 and following text).

The results in Figure 3.4b–d show that annotator competence and bias vary among annotators. Moreover, the figure shows that our model accurately estimates image difficulty, annotator competence, and annotator bias on data from real MTurk annotators.

Greeble Dataset: In the second experiment, annotators were shown pictures of "greebles" (see Figure 3.5) and were told that the greebles belonged to one of two classes. Some annotators were told that the two greeble classes could be discriminated by height, while others were told they could be discriminated by color (yellowish vs. green). This was done to explore the scenario in which annotators have different types of prior knowledge or abilities. We used a total of 200 images with 20 annotators

labeling each image. The height and color parameters for the two types of greebles were randomly generated according to Gaussian distributions with centers $(1, 1)$ and $(-1, -1)$, and standard deviations of 0.8.

The results in Figure 3.5 show that the model successfully learned two clusters of annotator decision surfaces, one (green) of which responds mostly to the first dimension of x_i (color) and another (red) responding mostly to the second dimension of x_i (height). These two clusters coincide with the sets of annotators primed with the two different attributes. Additionally, for the second attribute, we observed a few "adversarial" annotators whose labels tended to be inverted from their true values. This was because the instructions to our color annotation task were ambiguously worded, so that some annotators had become confused and had inverted their labels. Our model robustly handles these adversarial labels by inverting the sign of the $\hat{w}$ vector.

Waterbird Dataset: The greeble experiment shows that our model is able to segregate annotators looking for different attributes in images. To see whether the same phenomenon could be observed in a task involving images of real objects, we constructed an image dataset of waterbirds. We collected 50 photographs each of the bird species Mallard, American Black Duck, Canada Goose and Red-necked Grebe. In addition to the 200 images of waterbirds, we also selected 40 images without any birds at all (such as photos of various nature scenes and objects) or where birds were too small be seen clearly, making 240 images in total. For each image, we asked 40 annotators on MTurk if they could see a duck in the image (only Mallards and American Black Ducks are ducks). The hypothesis was that some annotators would be able to discriminate ducks from the two other bird species, while others would confuse ducks with geese and/or grebes.

Results from the experiment, shown in Figure 3.6, suggest that there are at least three different groups of annotators, those who separate: (1) ducks from everything else, (2) ducks and grebes from everything else, and (3) ducks, grebes, and geese from everything else; see numbered circles in Figure 3.6. Interestingly, the first group of annotators was better at separating out Canada geese than Red-necked grebes. This

may be because Canada geese are quite distinctive with their long, black necks, while the grebes have shorter necks and look more duck-like in most poses. There were also a few outlier annotators that did not provide answers consistent with any other annotators. This is a common phenomenon on MTurk, where a small percentage of the annotators will provide bad quality labels in the hope of still getting paid [SOJN08]. We also compared the labels predicted by the different models to the ground truth. Majority voting performed at 68.3% correct labels, GLAD at 60.4%, and our model performed at 75.4%.

3.7 Conclusions

We have proposed a Bayesian generative probabilistic model for the annotation process. Given only binary labels of images from many different annotators, it is possible to infer not only the underlying class (or value) of the image, but also parameters such as image difficulty and annotator competence and bias. Furthermore, the model represents both the images and the annotators as multidimensional entities, with different high level attributes and strengths respectively. Experiments with images annotated by MTurk workers show that indeed different annotators have variable competence level and widely different biases, and that the annotators' classification criterion is best modeled in multidimensional space. Ultimately, our model can accurately estimate the ground truth labels by integrating the labels provided by several annotators with different skills, and it does so better than the current state-of-the-art methods.

Besides estimating ground truth classes from binary labels, our model provides information that is valuable for defining loss functions and for training classifiers. For example, the image parameters estimated by our model could be taken into account for weighing different training examples, or, more generally, it could be used for a softer definition of ground truth. Furthermore, our findings suggest that annotators fall into different groups depending on their expertise and on how they perceive the task. This could be used to select annotators that are experts on certain tasks and

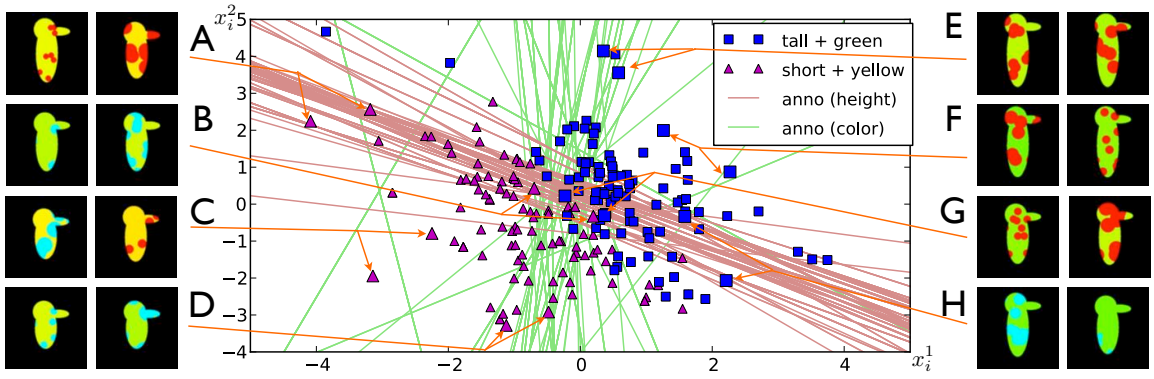

Figure 3.5: Estimated image parameters (symbols) and annotator decision planes (lines) for the greeble experiment. Our model learns two image parameter dimensions x_i^1 and x_i^2 which roughly correspond to color and height, and identifies two clusters of annotator decision planes, which correctly correspond to annotators primed with color information (green lines) and height information (red lines). On the left are example images of class 1, which are shorter and more yellow (red and blue dots are uncorrelated with class), and on the right are images of class 2, which are taller and more green. C and F are easy for all annotators, A and H are difficult for annotators that prefer height but easy for annotators that prefer color, D and E are difficult for annotators that prefer color but easy for annotators that prefer height, B and G are difficult for all annotators.

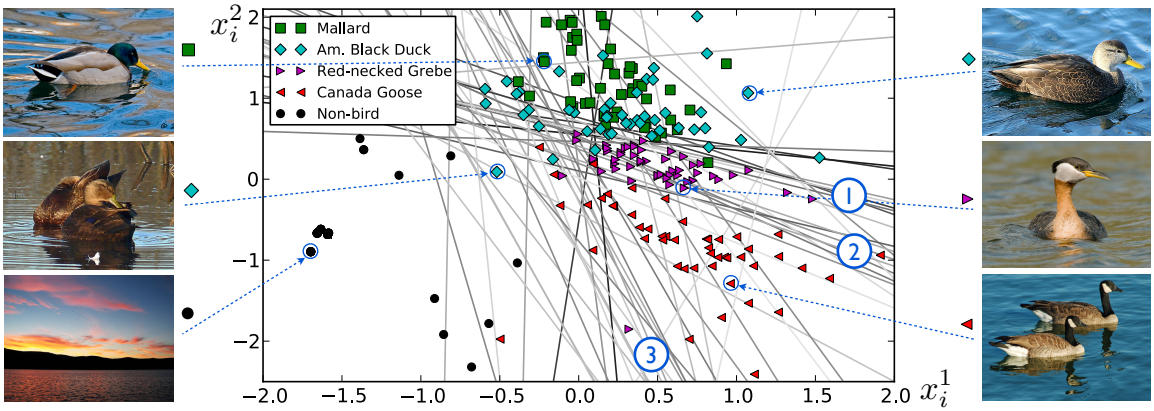

Figure 3.6: Estimated image and annotator parameters on the Waterbirds dataset. The annotators were asked to select images containing at least one "duck". The estimated x_i parameters for each image are marked with symbols that are specific to the class the image belongs to. The arrows show the x_i coordinates of some example images. The gray lines are the decision planes of the annotators. The darkness of the lines is an indicator of $\|w_j\|$: darker gray means the model estimated the annotator to be more competent. Notice how the annotators' decision planes fall roughly into three clusters, marked by the blue circles and discussed in Section 3.6.2.

to discover different schools of thought on how to carry out a given task.

Chapter 4

Confidence Labels

4.1 Abstract

Is it useful to ask crowdsourcing annotators to rate their level of confidence in a binary decision? How many levels of confidence should one use? What is the best way to come to an agreement with the annotators on what the levels of confidence mean? We explore these questions by means of a simple model of the annotation process, and four experiments in which hundreds of workers label complex synthetically generated images. We find analytically that a small number of confidence levels is typically sufficient, and we provide a closed-form expression for computing the most informative confidence thresholds. We show that it is possible to map such thresholds to rewards and penalties that annotators can experience during a training session. We observe experimentally that annotators will use confidence levels more consistently when trained with appropriate rewards and penalties, rather than using the popular Likert scale.

4.2 Introduction

Suppose that one wanted to label a large dataset with binary annotations, i.e. where the labels are either 'yes' or 'no'. One could crowdsource the task and determine the most likely label for each image by aggregating the opinion of multiple annotators, for example by majority voting. However, it is entirely possible that some of the

annotators may not be completely sure about the label that each image should be given. This may be because the annotators are confused, or because of some inherent ambiguity in the task. Thus, sometimes it may make sense to collect not only the annotators' labels, but also their level of certainty. So instead of answering 'yes' or 'no', the annotator would provide an *ordered categorical confidence label*, such as 'Definitely', 'Probably', 'Guessing', 'Probably not', or 'Definitely not'. Is there a principled way to think about this problem?

Our goal in this paper is to study three questions: (1) How do annotators perceive and make use of ordered categorical labels? We approach this question by proposing a probabilistic generative process for the labeling process and study its properties. (2) How can annotators optimally make use of the different label categories? We show how to use the labels in a way that reveals maximum information about the underlying ground truth class label. (3) How can we communicate to the annotators how to optimally make use of the labels? It has been noted that annotators in binary labeling experiments can learn optimal decision criteria quickly from maximizing their reward by incremental trial-and-error [NKP09]. We were curious to know whether this would be true for annotators who work with polytomous labels as in our case. We propose two methods for training the annotators, and show their effectiveness on MTurk annotators.

The paper is organized as follows: Section 4.3 summarizes related work on crowdsourcing and modeling annotators. Section 4.4 sets the stage and outlines the problem by exploring an annotation task carried out on MTurk. Section 4.5 proposes a generative probabilistic model for the annotation process and relates it to previous work. Section 4.6 studies a strategy for how to optimally make use of ordered categorical labels, and Section 4.7 describes how to effectively communicate this strategy to the annotators. Section 4.8 describes the results from experiments carried out on MTurk. Finally, conclusions and future directions are summarized in Section 4.9.

4.3 Related Work

Crowdsourcing has become a standard tool for annotating data datasets [vAMM+08, RTMF08, SF08, DDS+09, SOJN08, SPI08] and training online machine learning algorithms [CBK+10, VG11]. There has been various works on crowdsourcing different kinds of annotations, including binary and discrete labels [WRW+09, WP10, WBBP10], continuous labels [SF08, WP10], and naming [SP08]. Unlike much of the previous work, we do not present a method for merging annotations. Instead, we focus on how to influence annotators to make use of ordered categorical labels in an effective manner.

Modeling the labeling process wherein one or more annotators provide response labels for a latent variable has a long history. One well known model is that of [Ras80] which models the labels by a logistic regression-like model. This is in turn related to Item Response Theory (IRT) [Lor80]. [DS79] is a method that is often used for estimating annotator bias for polytomous labels by modeling the annotators as confusion matrices. Recently, there has also been some work on considering label difficulty in the binary labeling process [WRW+09, WBBP10].

One of the first surveys on using ordered categorical labels was that of Likert [Lik32], which suggested the 5-point "Likert-scale." There has been some experimental work of applying information theory to study the effectiveness of different rating scales [Cox80], but we are not aware of any work that has studied a model like ours. [Lin02] lists heuristic guidelines for using more or less categories, but provides little theoretical analysis.

4.4 An Exploratory Experiment

To investigate how annotators on MTurk make use of ordered categorical labels, we devised an exploratory experiment where the annotators were asked to annotate synthetically generated images (see Figure 4.1a-b). The images were generated by placing a 7×7 perturbed grid of dots of different contrast on a gray background.

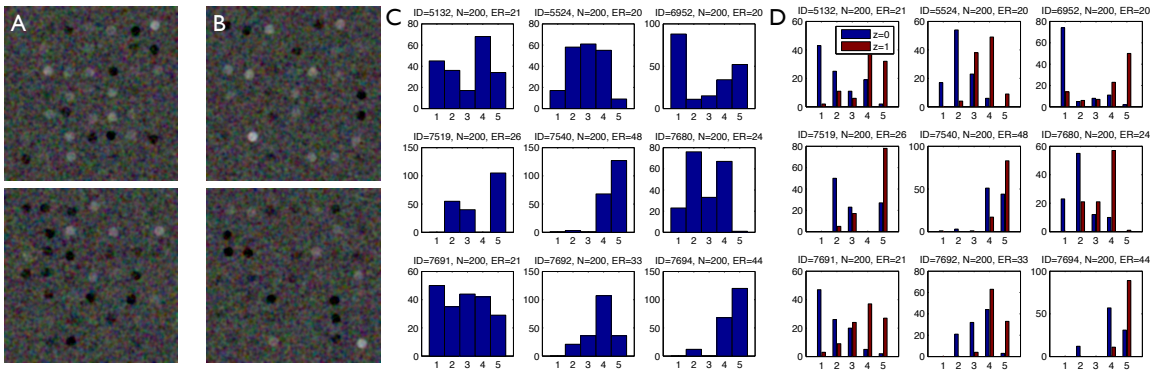

Figure 4.1: The exploratory experiment. A-B: Annotators were shown synthetically generated images with dots of different contrast. For some of the images, they were told that the image showed signs of radioactivity. The two images in panel (A) were said to show no signs of radioactivity while the images in panel (B) were said to have radioactivity (indicated by the bright dots in the lower-left and lower-right corners, respectively). C: Distribution of label choices obtained from the nine workers that labeled the most images. The x-axis shows the label and the y-axis is the number of images that were given the label. The *ID* is the identity of the annotator, N is the number of images the annotator labeled, and *ER* is the total error rate in percent. D: The distribution of labels for images of the background ($z = 0$) and foreground ($z = 1$) classes.

Half of the images belonged to the *background class*. The contrasts of the dots in the background class were drawn from a Normal distribution (see Figure 4.1a). The other half of the images belonged to the *foreground class*. In these images, the contrast for all but one of the dots were drawn from the same distribution as for the background class. The remaining dot, randomly chosen amongst the 49 available ones, was given a contrast drawn from a Normal distribution with a higher mean brightness (see Figure 4.1b). All images were corrupted by slightly smoothed Gaussian noise.

We explained to the annotators that the task was an experiment in visual perception, and that it simulated the task of "finding radioactivity in images." They were required to go through a training session with 100 images. For each image, they had to answer 'Yes' or 'No' on the question of whether they believed there was radioactivity in the image. After each answer, they were given instant feedback on whether their answer was correct and, if the image was from the foreground class, were shown an arrow pointing at the dot with contrast drawn from the foreground distribution. After the training session, the annotators were tasked with labeling images without getting any feedback. For these images, the annotators were asked how much they agreed with the statement that "There is radioactivity in this image." They had to answer using a standard 5-point Likert scale with labels $\{Strongly\ agree,\ Agree,\ Neither\ agree\ nor\ disagree,\ Disagree,\ Strongly\ disagree\}$. The annotators could annotate up to 200 images in batches of 10 images at a time.

Figure 4.1c shows the distributions of labels provided by the nine workers that annotated most images. As can be seen from the histograms, the annotators vary substantially in how they distribute their labels. Some annotators make use of the labels in more or less equal proportions, while others are polarized in how they pick them. Figure 4.1d shows a similar variety when the labels are separated into two histograms, one for the foreground and one for the background class. Qualitatively, there are a few kinds of different annotators: (1) The meek annotators that use the middle labels when unsure, and only use extreme labels when they are very certain of their answers. (2) The bold annotators that have strong opinions and rarely make use of the middle labels. (3) The well-rounded annotators that make equal use of all

labels.

4.5 Model of the Labeling Process

We follow [WBBP10] and model the annotators and labels in a signal detection framework. Every image, indexed by i, has a ground truth binary target value z_i. For example, in the radioactivity task described in Section 4.4 $z_i = 1$ means that radioactivity is present and $z_i = 0$ means it is not. We assume that image i has a "signal" x_i that is indicative of z_i. The signal is assumed to be produced by the generative process $p(x_i \mid z_i)$, which is modeled as a Normal distribution with mean μ_{z_i} and variance $\Sigma^2_{z_i}$ (see Figure 4.2a),

$$p(x \mid z) = \mathcal{N}(x \mid \mu_z, \Sigma^2_z), \tag{4.1}$$

where subscripts have been dropped for brevity, as we will henceforth consider a single image. We assume a Bernoulli prior on z, set to $p(z = 1) = 0.5$ in the examples throughout the paper.

The image is observed by m annotators, indexed by j, providing labels $L_m = \{l_1, l_2, \ldots, l_m\}$, where l_j is the label provided by annotator j. Each label can take an integer value, $l_j \in \{1, 2, \ldots, T\}$. The number of possible label choices T depends on the task. For example, $T = 2$ for binary tasks and $T = 5$ for the 5-point Likert scale. As in [WBBP10], we assume that annotator j does not observe x directly, but instead sees a signal y_j corrupted by Gaussian noise (see the lower part of Figure 4.2a), i.e. $p(y_j|x) = \mathcal{N}(y_j \mid x, \sigma_j^2)$. Thus, σ_j indicates how clearly annotator j can perceive the signal. The annotators make their decision on the labels l_i based on a vector of thresholds, $\boldsymbol{\tau}_j = (\tau_{j,0}, \tau_{j,1}, \ldots, \tau_{j,T})$, where $\tau_{j,0} = -\infty$ and $\tau_{j,T} = \infty$ (see Figure 4.2b). The probability of the label l_j is given by the indicator function, $p(l_j = t \mid \boldsymbol{\tau}_j) = \mathbf{1}(\tau_{j,t-1} \leq y_j \leq \tau_{j,t})$. To reduce the complexity in the analysis that follows we assume the annotators share the same parameters, i.e. $\sigma_j = \sigma$ and $\boldsymbol{\tau}_j = \boldsymbol{\tau}$.

We can compute the probability of label l_j given an image signal x directly by

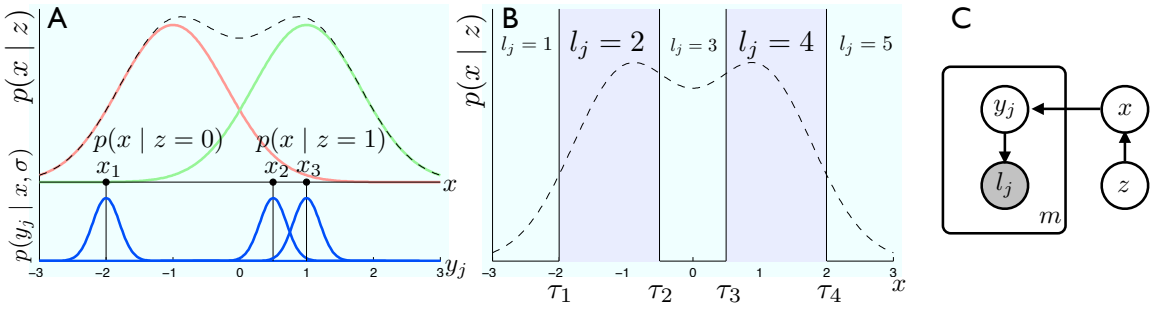

Figure 4.2: Model of the labeling process. A: The top section of the plot shows the conditional distributions $p(x \mid z)$ for $z = 0$ (red) and $z = 1$ (green). The dashed line is the sum of the two distributions. Three image signals (x_1, x_2, x_3) have been sampled from the distributions and are shown as circular black markers. The signal that annotator j sees is y_j, which is sampled from a Normal distribution (shown as a blue curve) centered on the x_i value. B: For an ideal annotator (one with $\sigma = 0$) the label l_j is determined by comparing the image signal x with the thresholds $\tau_1, \ldots, \tau_4$. If it falls between τ_{t-1} and τ_t, it gets assigned label $l_j = t$. For annotators with $\sigma_j > 0$, the process is the same, albeit with some noise (see Section 4.5 for details). C: Graphical representation of the generative process by which the labels are created. Given the class z, an image-specific signal x is generated. Each annotator j sees a different corrupted version y_j of x and assigns label l_j by comparing y_j with the vector of thresholds $\boldsymbol{\tau}$.

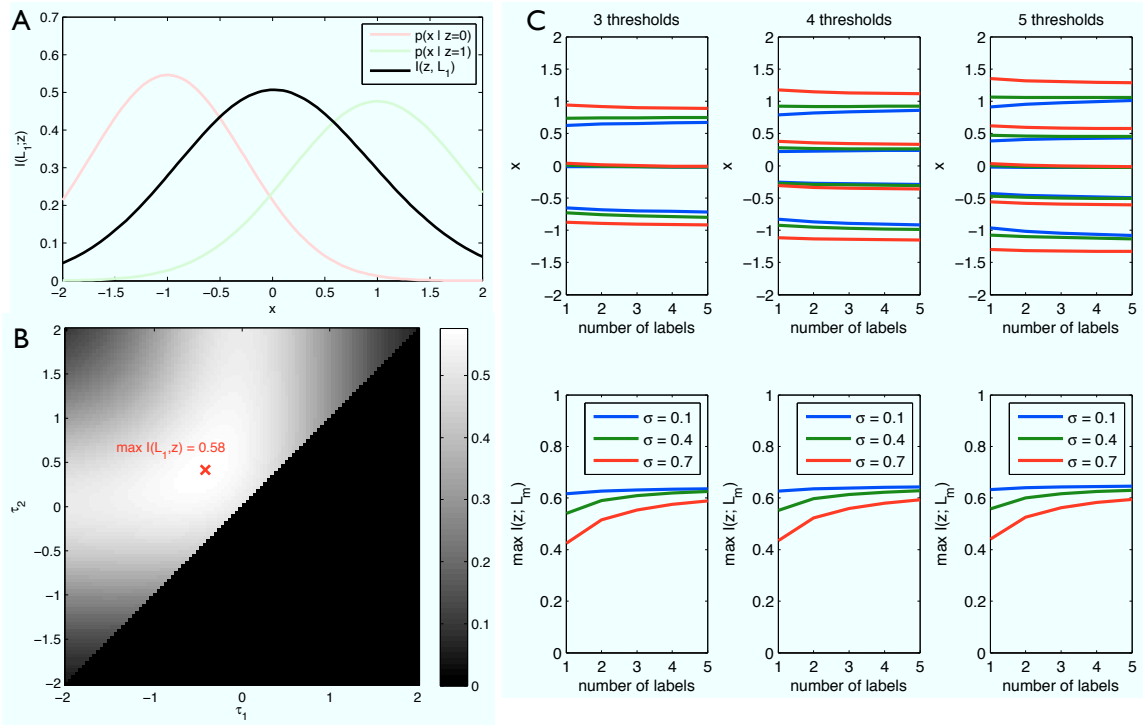

Figure 4.3: Making labels informative by maximizing mutual information. A: Shows the mutual information $I(z; L_m)$ between a label from a single annotator, $L_1 = \{l_1\}$ and the ground truth label z as τ is varied from -2 to 2. For comparison the class-conditional distributions $p(x \mid z)$ are also shown. $I(z; L_m)$ is maximized when τ_1 is close to zero. B: Heat map showing the value of $I(z; L_m)$ when using two thresholds and a single label. τ_2 is constrained to be greater than τ_1. The $\boldsymbol{\tau}$ for which $I(z; L_m)$ is maximized is shown by the red cross. C: Distribution of the τ_t for different lengths of the vector $\boldsymbol{\tau}$. The bottom plot show the maximum mutual information achieved at different m, i.e. number of annotators that labeled the image, for 3, 4, and 5 thresholds. The curves of different colors represent different values of σ. For small values of σ, the addition of more annotators does little to increase $I(z; L_m)$, but for larger values of σ (less accurate annotators), the number of annotators has a big impact. The top plots show how the optimal thresholds are placed.

integrating out y_j,

$$p(l_j = t \mid x) = \int_{\tau_{t-1}}^{\tau_t} \mathcal{N}(y_j \mid x, \sigma^2)\, dy_j = \Phi(\tau_t \mid x, \sigma^2) - \Phi(\tau_{t-1} \mid x, \sigma^2), \tag{4.2}$$

where $\Phi(\tau \mid x, \sigma^2) = \int_{-\infty}^{\tau} \mathcal{N}(y \mid x, \sigma^2)\, dy$ is the cumulative Normal distribution. The full generative process is described graphically in Figure 4.2c.

As we shall see in the next section, the posterior on z is of special interest when computing how much information the labels provide about the underlying class label z. The posterior can be computed using Bayes' rule, $p(z \mid L_m) = p(L_m \mid z)p(z)/p(L_m)$ where,

$$p(L_m \mid z) = \int_{-\infty}^{\infty} p(L_m, x \mid z)\, dx = \int_{-\infty}^{\infty} \left(\prod_{j=1}^{m} p(l_j \mid x) \right) p(x \mid z)\, dx. \tag{4.3}$$

The last equality comes from the assumption that the annotators provide the labels independently of each other given that they observe the same signal x.

4.6 Informative Labels

According to the model presented in Section 4.5, the distribution of the labels that the annotators provide depends on the vector of thresholds $\boldsymbol{\tau}$. A natural question to ask is if there are some values of $\boldsymbol{\tau}$ that are better than others? There are two dimensions to this question: (1) How many labels T should the annotators have to choose from. (2) Given T, what thresholds should the annotators use?

One way to approach these questions is to treat the labeling process as an information theoretic channel. In that case, the labels L_m obtained from the annotators provide information about the underlying class distribution $p(z)$. Using Figure 4.2a as an example, if the labels obtained from two annotators for an image are $L_m = \{1, 1\}$, then it is quite likely that $z = 0$. The extent to which knowing L_m reduces the

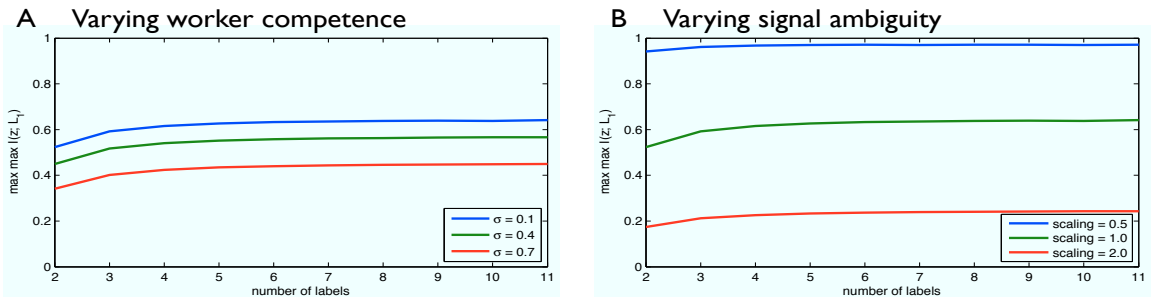

Figure 4.4: How much choice should annotators have? A: Shows the maximum mutual information for different worker competences σ_j as the number of label choices increases from 2 to 11. B: Show the same, but varies the scaling of the Σ_z used in Figure 4.3. A scaling of 2 means the original Σ_z's are doubled, thus making the task more ambiguous and uncertain. When the Σ_z's is halved, the task gets easier and it is almost possible to extract a full bit from a single annotator.

uncertainty about z is given by the mutual information, defined as:

$$I(z;\, L_m) = H(L_m) - H(L_m \mid z), \tag{4.4}$$

where $H(\cdot)$ denotes the entropy. Figure 4.3a-b shows how $I(z;\, L_m)$ varies with $\boldsymbol{\tau}$ for $T = 2$ and $T = 3$. For example, in the case of $T = 2$, the mutual information approaches zero when τ_1 is very positive or very negative, since at that point the labels provided by the annotators will always be the same, independently of the underlying ground truth class. In this particular example, the maximum $I(z;\, L_m)$ is achieved when τ_1 is very close to zero. This example suggests that one principle for choosing $\boldsymbol{\tau}$ is such that the mutual information is maximized:

$$\boldsymbol{\tau}^\star = \arg\max_{\boldsymbol{\tau}} I(z; L_m \mid \boldsymbol{\tau}), \tag{4.5}$$

where the notation $I(z; L_m \mid \boldsymbol{\tau})$ is used to denote the value of $I(z; L_m)$ as a function of $\boldsymbol{\tau}$.

In order to efficiently maximize the mutual information using gradient ascent-based methods, the gradient $\nabla I(z; L_m \mid \boldsymbol{\tau})$ with respect to $\boldsymbol{\tau}$ needs to be computed. This can be done by noting that the partial derivative of Equation 4.3 with respect

to τ_t is given by,

$$\frac{\partial}{\partial \tau_t} p(L_m \mid z) = \int_{-\infty}^{\infty} \frac{\partial}{\partial \tau_t} \left(\prod_{j=1}^{m} p(l_j \mid x) \right) p(x \mid z)\, dx. \tag{4.6}$$

This expression can be evaluated by taking advantage of that the labels provided by the annotators are exchangeable. That means that they can be grouped according to the values of the labels. This, if $n(t)$ is used to denote the number of labels in the vector L_m with $l_j = t$, i.e. $n(t) = |\{l_j \in L_m \mid l_j = t\}|$, then

$$\frac{\partial}{\partial \tau_t} \left(\prod_{j=1}^{m} p(l_j \mid x) \right) = \frac{\partial}{\partial \tau_t} \left(\prod_{k=1}^{T} p(l_j = k \mid x)^{n(k)} \right) = D_t(x) \prod_{k \notin \{t, t+1\}} p(l_j = k \mid x)^{n(k)} \tag{4.7}$$

where

$$D_t(x) = n(t) \frac{\partial p(l_j = t \mid x)}{\partial \tau_t} p(l_j = t \mid x)^{n(t)-1} p(l_j = t+1 \mid x)^{n(t+1)} \tag{4.8}$$

$$+ n(t+1) \frac{\partial p(l_j = t+1 \mid x)}{\partial \tau_t} p(l_j = t \mid x)^{n(t)} p(l_j = t+1 \mid x)^{n(t+1)-1} \tag{4.9}$$

and $\frac{\partial}{\partial \tau_t} p(l_j = t \mid x) = \mathcal{N}(\tau_t \mid x, \sigma)$ and $\frac{\partial}{\partial \tau_t} p(l_j = t+1 \mid x) = -\mathcal{N}(\tau_t \mid x, \sigma)$.

Using these derivatives it is possible to maximize the mutual information with respect to $\boldsymbol{\tau}$. However, since computing $I(z; L_m \mid \boldsymbol{\tau})$ involves an expectation over all possible values of (z, L_m), and thus is exponential in m, it is time-consuming to compute for more than a few annotators.

Figure 4.3c (top) shows the optimal placement of the thresholds $\boldsymbol{\tau}$ for different values of T and different number of labels m provided by the annotators. The colors of the curves indicate the σ of the annotators. For large σ, i.e. less accurate annotators, the thresholds are pushed further from the origin. Figure 4.3c (bottom) shows the values of $I(z; L_m \mid \boldsymbol{\tau}^\star)$ at the optimal thresholds $\boldsymbol{\tau}^\star$ as the number of labels m obtained increases. Intuitively, when σ is small, only 1-2 labels are needed to reach a high mutual information, while when σ is larger more labels are needed.

Figure 4.4 shows how many label choices are needed to reach a high mutual in-

formation. From Figure 4.4a, it is clear that the annotator accuracy, indicated by σ, is a limiting factor on how much information the annotators can provide. In Figure 4.4b, the Σ_z of the underlying class conditional densities $p(x \mid z)$, as defined in Equation 4.1, are scaled by 0.5, 1, and 2. When scaled to half their initial width, such that the overlap between the distributions is small, it is possible to almost reach the upper bound of 1 bit of mutual information. The main message of Figure 4.4 is there is little value from using more than 4–5 choices, something also seen in experimental studies [Cox80].

4.7 Communicating Thresholds

Given knowledge about $p(z)$, $p(x \mid z)$ and σ, we can find the optimal set of thresholds $\boldsymbol{\tau}$ for any number m of labels. However, the problem of getting the annotators to use these thresholds remains. The experiment in Section 4.4 showed that without guidance of how to interpret the labels, the annotators will vary wildly in how they distribute their labels. Some will use only the extreme values, while others will often resort to middle-valued labels when uncertain. In this section, we suggest two methods for communicating the thresholds to the annotators: (1) using probabilities and (2) using rewards.

The first method is to communicate the thresholds using probabilities. Given an image with signal x, the annotators are asked to choose the labels based on their internal estimate $p_1 = p(z = 1 \mid x)$. For example, if $T = 3$, they are asked to pick $l_j = 1$ if $p_1 < p(z = 1 \mid x = \tau_1)$, $l_j = 2$ if $p(z = 1 \mid x = \tau_1) \leq p_1 < p(z = 1 \mid x = \tau_2)$ and $l_j = 3$ if $p_1 \geq p(z = 1 \mid x = \tau_2)$. Thus, the instructions to the annotator might say: "Choose label 3 if you think that the likelihood of there being radioactivity in the image is greater than 90%." This requires that the annotators learn $p(z)$ and $p(x \mid z)$ so that they can estimate p_1 using Bayes' rule, $p(z = 1 \mid x) = p(x \mid z)\, p(z)/p(x)$. We argue that the annotators can learn these distributions by allowing them to train on images where they know the ground truth value z_i for each image. We suggest a setting for training the annotators in Section 4.8.

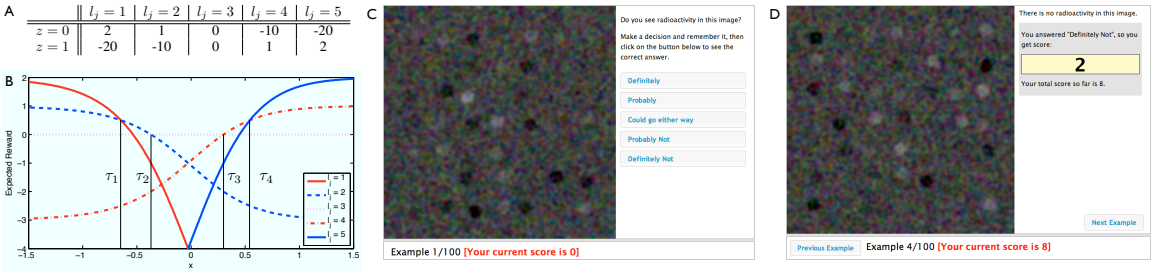

Figure 4.5: Communicating using reward matrices. A: Sample reward matrix used in Section 4.8. B: The expected reward when using different labels l_j for different values of x. A rational annotator would choose the thresholds where the maximally rewarding label l_j switches. C: Interface used to train the annotators in the RewardOpt and RewardPol experiments. D: Given the response and the ground truth class of the image, the annotator is rewarded a score for the training image. He/she also sees the total score accumulated so far.

The second method is to communicate the thresholds through costs and rewards. The answers of the annotators will benefit us if they are correct and will cost us if they are incorrect. If levels of uncertainty are allowed, it will be best that the annotators will give us the right answer with the highest level of confidence, but if the wrong answer is given it should carry the least amount of confidence. A rational way of designing the task is to pass on these benefits and costs to the annotators. For example, one could have experts annotate a small subset of the images, thus providing a ground truth, and compute the annotators' compensation based on their performance on these images with costs and benefits specified by a reward matrix $\mathbf{R}$. The entry $r_{z,t-1}$ of $\mathbf{R}$ is the reward given to annotator for label $l_j = t$ when the ground truth class is z. An example reward matrix is shown in Figure 4.5a.

How would a rational annotator make use of this information? Again, assuming that the annotator has had the opportunity to estimate $P(z = 1|x)$, then the optimal strategy for the annotator would be to compute the rewards he expects when he observes x and gives each possible answer l_j (denoted here with $R(l_j|x)$). The best answer one should give when observing a given x is l_j^* that maximizes the reward

$R(l_j|x)$:

$$\begin{aligned} R(l_j|x) &= r_{0,l_j-1}P(z=0|x) + r_{1,l_j-1}P(z=1|x) && (4.10)\\ &= r_{0,l_j-1} + P(z=1|x)\left(r_{1,l_j-1} - r_{0,l_j-1}\right) && (4.11)\\ l_j^* &= \arg\max_{l_j} R(l_j|x) && (4.12) \end{aligned}$$

An example of this computation can be seen in Figure 4.5b for the reward matrix in the Figure 4.5a. Each line shows the expected reward when answering a particular label l_j for a range of x-values. A rational annotator would place the thresholds $\boldsymbol{\tau}$ at values of x where the label l_j for which the expected reward is maximized switches. At these points, we have the constraints,

$$R(l_j = t \mid x = \tau_t) = R(l_j = t+1 \mid x = \tau_t). \tag{4.13}$$

Thus, given a reward matrix R, it is possible to find the vector of thresholds $\boldsymbol{\tau}$ that maximizes the expected reward.

If we already know the thresholds $\boldsymbol{\tau}$, how can we pick a reward matrix R such that an annotator would use those thresholds? We can encode the constraints from Equation 4.13 in a $(T-1) \times (2T)$ matrix $\mathbf{A}$ and write down a system of linear equations,

$$\mathbf{Ar} = \mathbf{0} \quad \text{where} \quad a_{uv} = \begin{cases} 1 - p_1(t) & \text{if } u = t-1 \text{ and } v = t-1, \\ p_1(t) - 1 & \text{if } u = t-1 \text{ and } v = t, \\ p_1(t) & \text{if } u = t-1 \text{ and } v = T, \\ -p_1(t) & \text{if } u = t-1 \text{ and } v = T+t, \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

The vector $\mathbf{r} = \text{vec}(\mathbf{R}^{\mathrm{T}})$ is the rows of $\mathbf{R}$ concatenated into a column vector, and $p_1(t) = p(z = 1 \mid x = \tau_t)$. This is an underdetermined problem, and thus the null space $\mathbf{K}$ of $\mathbf{A}$ is the space of possible reward matrices that will work with the thresholds $\boldsymbol{\tau}$. Which of the infinitely many reward matrices should one choose? One

method is to find a reward matrix $\hat{\mathbf{R}}$ that is close to $\mathbf{R}$ using least squares, i.e. $\hat{\mathbf{r}} = \mathbf{K}\hat{\boldsymbol{\beta}}$ where $\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{r} - \mathbf{K}\boldsymbol{\beta}\|^2$ and $\hat{\mathbf{r}} = \text{vec}(\hat{\mathbf{R}}^{\text{T}})$.

4.8 Experiments

In order to investigate how well annotators could make use of the labels, we prepared four experiments on MTurk named *Likert*, *Percent*, *RewardOpt*, and *RewardPol*. Our goal was to answer the following three questions: (1) Given few instructions about how to use the available label choices, how will annotators distribute their thresholds? (2) Can we communicate thresholds to annotators through the probabilities? (3) Can we influence the placement of the annotators thresholds by training them with different reward matrices?

The experimental setup was near identical to that described in Section 4.4, but with some small but important differences for three out of the four experiments. Since we knew the underlying feature (contrast of brightest dot) and signal distribution used to generate the images, we could use the reward matrix from Figure 4.5a and the methods from Section 4.6 to find the optimal thresholds $\boldsymbol{\tau}^{\star}$. Then, for the Percent experiment, we converted the thresholds to percentages, as described in Section 4.7, and asked the annotators to choose between five ranges of probabilities for each image[1]. In the RewardOpt experiment, instead of giving the percentages, we used the training session to train the annotators using the reward matrix. We used the interface shown in Figure 4.5b-c to give a score based on the answer the annotator provided (the annotators were also told that the four annotators achieving the highest score on the test images would be given a $5 reward). The scores were only shown to the annotators in the training, and not during the testing. The RewardPol experiment was identical to RewardOpt, except that we used a different reward matrix[2] that encouraged annotators to only use the extreme (polarized) labels, $l_j = 1$ and $l_j = 5$. The Likert experiment was identical to the experiment in Section 4.4.

[1]Labels $l_j = 1, \ldots, 5$ corresponded to the ranges 0-10%, 10-30%, 30-70%, 70-90%, and 90-100%.
[2]We used $\mathbf{R} = [[10, 1, 0, -1, -10], [-10, -1, 0, 1, 10]]$, which enforces $\tau_1 = \tau_2 = \tau_3 = \tau_4$.

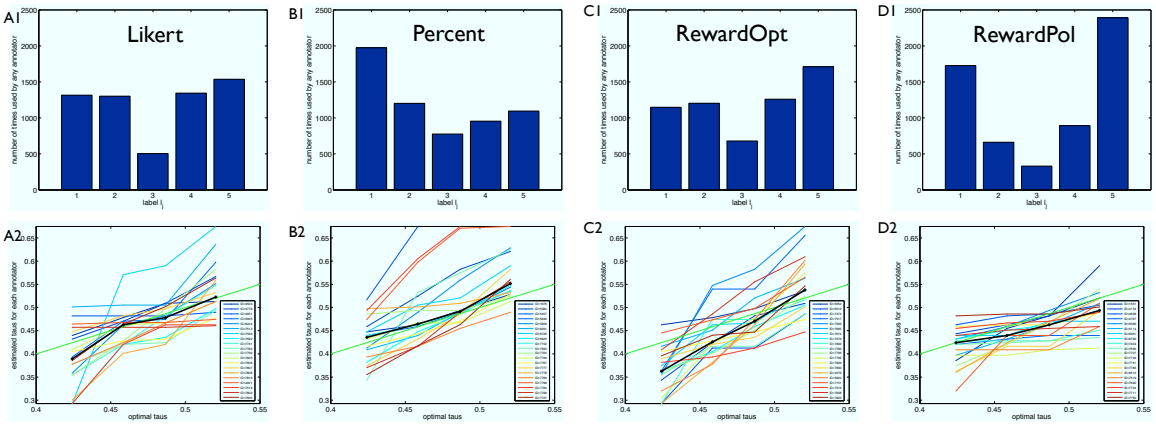

Figure 4.6: Experiments. Top row shows the number of images receiving each label for each of the four experiments. The bottom row shows the estimated $\boldsymbol{\tau}_j$ for the 20 annotators that labeled most images, versus the optimal thresholds as computed by the methods in Section 4.6. See Section 4.8 for a discussion.

The results of the experiments are shown in Figure 4.6. The top row in the figure shows how the annotators distribute the images amongst the labels. Using the 5-point Likert scale, the annotators tend to avoid $l_j = 3$. However, when guided to use the optimal thresholds (Percent and RewardOpt), the annotators make more use of the middle labels. Histogram D1 also shows that the reward matrix can be used to make annotators avoid the middle labels.

The bottom row in Figure 4.6 shows how close the thresholds that the annotators actually use are to the optimal thresholds. The thresholds that annotator j make use of can be computed using a maximum likelihood estimator by noting that the the labels l_{ij} provided by the annotator for different images signals x_i are independent given σ and $\boldsymbol{\tau}_j$ (we use $\sigma = 0.01$). Qualitatively, we can see that for the Likert experiment, we have two kinds of annotators: the ones that match the optimal thresholds quite well, and the ones that essentially treats the task as a binary task, only using the extreme labels (shown as horizontal lines in panel A2). However, when communicating the thresholds explicitly in the Percent and RewardOpt experiments, all annotators spread out their thresholds to match the optimal thresholds better. In the RewardPol experiments, most annotators ignore the middle labels, appearing as horizontal lines in panel D2, just as some annotators did in the Likert experiment.

4.9 Discussion and Conclusion

We explored the question of how to use confidence levels in crowdsourced annotation. Using a simple model of the annotation process we found in that, in realistic scenarios, a small number (3-5) of confidence levels is sufficient to approach the maximum mutual information between the annotators' labels and the underlying ground truth. Using the same technique, we find that it is possible to compute the optimal thresholds for using a given set of discrete confidence labels.

We highlighted the challenge that an experimenter faces of coming to an agreement with the annotators on what is the probabilistic meaning of a given set of confidence labels. Ideally, the experimenter would have means to 'program' the annotators to use the optimal set of confidence thresholds for a given task. We proposed two techniques for doing so: explicitly, using a list of probability levels, and implicitly, via training with a reward-cost matrix that, we show, may be computed from the thresholds. We find that both methods work better than the traditional Likert scale.

Chapter 5

Clustering using Pairwise Labels

5.1 Abstract

Is it possible to crowdsource categorization? Amongst the challenges: (a) each worker has only a partial view of the data, (b) different workers may have different clustering criteria and may produce different numbers of categories, (c) the underlying category structure may be hierarchical. We propose a Bayesian model of how workers may approach clustering and show how one may infer clusters / categories, as well as worker parameters, using this model. Our experiments, carried out on large collections of images, suggest that Bayesian crowdclustering works well and may be superior to single-expert annotations.

5.2 Introduction

Outsourcing information processing to large groups of anonymous workers has been made easier by the internet. *Crowdsourcing* services, such as Amazon's Mechanical Turk, provide a convenient way to purchase *Human Intelligence Tasks* (HITs). Machine vision and machine learning researchers have begun using crowdsourcing to label large sets of data (e.g., images and video [SF08, VG11, WBBP10]) which may then be used as training data for AI and computer vision systems. In all the work so far *categories* are defined by a scientist, while *categorical labels* are provided by the workers.

Can we use crowdsourcing to *discover* categories? I.e., is it possible to use crowdsourcing not only to *classify* data instances into established categories, but also to *define the categories* in the first place? This question is motivated by practical considerations. If we have a large number of images, perhaps several tens of thousands or more, it may not be realistic to expect a single person to look at all images and form an opinion as to how to categorize them. Additionally, individuals, whether untrained or expert, might not agree on the criteria used to define categories and may not even agree on the number of categories that are present. In some domains unsupervised clustering by machine may be of great help; however, unsupervised categorization of images and video is unfortunately a problem that is far from solved. Thus, it is an interesting question whether it is possible to collect and combine the opinion of multiple human operators, each one of which is able to view a (perhaps small) subset of a large image collection.

We explore the question of crowdsourcing clustering in two steps: (a) Reduce the problem to a number of independent HITs of reasonable size and assign them to a large pool of human workers (Section 5.3). (b) Develop a model of the annotation process, and use the model to aggregate the human data automatically (Section 5.4) yielding a partition of the dataset into categories. We explore the properties of our approach and algorithms on a number of real world data sets, and compare against existing methods in Section 5.5.

5.3 Eliciting Information from Workers

How shall we enable human operators to express their opinion on how to categorize a large collection of images? Whatever method we choose, it should be easy to learn and it should be implementable by means of a simple graphical user interface (GUI). Our approach (Figure 5.1) is based on displaying small subsets of M images and asking workers to group them by means of mouse clicks. We provide instructions that may cue workers to certain attributes but we do not provide the worker with category definitions or examples. The worker groups the M items into clusters of

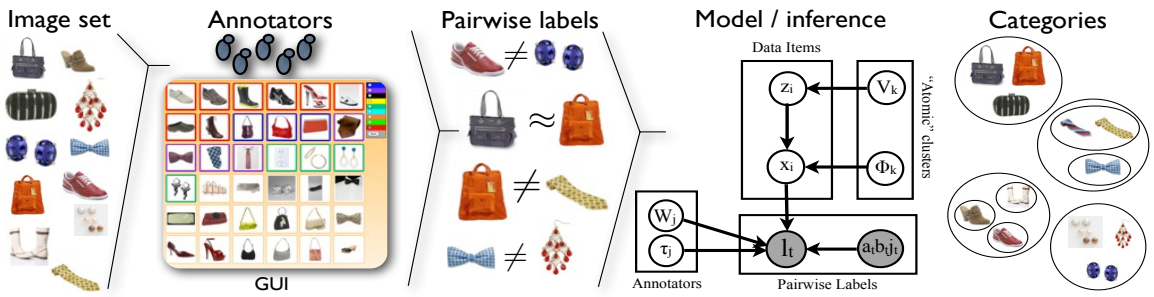

Figure 5.1: Schematic of Bayesian crowdclustering. A large image collection is explored by workers. In each HIT (Section 5.3), the worker views a small subset of images on a GUI. By associating (arbitrarily chosen) colors with sets of images the worker proposes a (partial) local clustering. Each HIT thus produces multiple binary pairwise labels: each pair of images shown in the same HIT is placed by the worker either in the same category or in different categories. Each image is viewed by multiple workers in different contexts. A model of the annotation process (Sec. 5.4.1) is used to compute the most likely set of categories from the binary labels. Worker parameters are estimated as well.

his choosing, as many as he sees fit. An item may be placed in its own cluster if it is unlike the others in the HIT. The choice of M trades off between the difficulty of the task (worker time required for a HIT increases super-linearly with the number of items), the resolution of the images (more images on the screen means that they will be smaller), and contextual information that may guide the worker to make more global category decisions (more images give a better context, see Section 5.5.1.) Partial clusterings on many M-sized subsets of the data from many different workers are thus the raw data on which we compute clustering.

An alternative would have been to use pairwise distance judgments or three-way comparisons. A large body of work exists in the social sciences that makes use of human-provided similarity values defined between pairs of data items (e.g., Multidimensional Scaling [Kru64].) After obtaining pairwise similarity ratings from workers, and producing a Euclidean embedding, one could conceivably proceed with unsupervised clustering of the data in the Euclidean space. However, accurate distance judgments may be more laborious to specify than partial clusterings. We chose to explore what we can achieve with partial clusterings alone.

We do not expect workers to agree on their definitions of categories, or to be

consistent in categorization when performing multiple HITs. Thus, we avoid explicitly associating categories across HITs. Instead, we represent the results of each HIT as a series of $\binom{M}{2}$ binary labels (see Figure 5.1). We assume that there are N total items (indexed by i), J workers (indexed by j), and H HITs (indexed by h). The information obtained from workers is a set of binary variables $\mathcal{L}$, with elements $l_t \in \{-1, +1\}$ indexed by a positive integer $t \in \{1, \ldots, T\}$. Associated with the t-th label is a quadruple (a_t, b_t, j_t, h_t), where $j_t \in \{1, \ldots, J\}$ indicates the worker that produced the label, and $a_t \in \{1, \ldots, N\}$ and $b_t \in \{1, \ldots, N\}$ indicate the two data items compared by the label. $h_t \in \{1, \ldots, H\}$ indicates the HIT from which the t-th pairwise label was derived. The number of labels is $T = H\binom{M}{2}$.

Sampling Procedure We have chosen to structure HITs as clustering tasks of M data items, so we must specify them. If we simply separate the items into disjoint sets, then it will be impossible to infer a clustering over the entire data set. We will not know whether two items in different HITs are in the same cluster or not. There must be some overlap or redundancy: data items must be members of multiple HITs.

In the other extreme, we could construct HITs such that each pair of items may be found in at least one HIT, so that every possible pairwise category relation is sampled. This would be quite expensive for large number of items N, since the number of labels scales asymptotically as $T \in \Omega(N^2)$. However, we expect a noisy transitive property to hold: if items a and b are likely to be in the same cluster, and items b and c are (not) likely in the same cluster, then items a and c are (not) likely to be in the same cluster as well. The transitive nature of binary cluster relations should allow sparse sampling, especially when the number of clusters is relatively small.

As a baseline sampling method, we use the random sampling scheme outlined by Strehl and Ghosh [SG02] developed for the problem of *object distributed* clustering, in which a partition of a complete data set is learned from a number of clusterings restricted to subsets of the data. (We compare our aggregation algorithm to this work in Section 5.5.) Their scheme controls the level of sampling redundancy with a single parameter V, which in our problem is interpreted as the expected number of HITs to

which a data item belongs.

The N items are first distributed deterministically among the HITs, so that there are $\lceil \frac{M}{V} \rceil$ items in each HIT. Then the remaining $M - \lceil \frac{M}{V} \rceil$ items in each HIT are filled by sampling without replacement from the $N - \lceil \frac{M}{V} \rceil$ items that are not yet allocated to the HIT. There are a total of $\lceil \frac{NV}{M} \rceil$ unique HITs. We introduce an additional parameter R, which is the number of different workers that perform each constructed HIT. The total number of HITs distributed to the crowdsourcing service is therefore $H = R\lceil \frac{NV}{M} \rceil$, and we impose the constraint that a worker can not perform the same HIT more than once. This sampling scheme generates $T = R\lceil \frac{NV}{M} \rceil \binom{M}{2} \in O(RNVM)$ binary labels.

With this exception, we find a dearth of ideas in the literature pertaining to sampling methods for distributed clustering problems. Iterative schemes that adaptively choose maximally informative HITs may be preferable to random sampling. We are currently exploring ideas in this direction.

5.4 Aggregation via Bayesian Crowdclustering

There is an extensive literature in machine learning on the problem of combining multiple alternative clusterings of data. This problem is known as consensus clustering [MTMG03], clustering aggregation [GMT07], or cluster ensembles [SG02]. While some of these methods can work with partial input clusterings, most have not been demonstrated in situations where the input clusterings involve only a small subset of the total data items ($M << N$), which is the case in our problem.

In addition, existing approaches focus on producing a single "average" clustering from a set of input clusterings. In contrast, we are not merely interested in the average clustering produced by a crowd of workers. Instead, we are interested in understanding the ways in which different individuals may categorize the data. We seek a master clustering of the data that may be combined in order to describe the tendencies of individual workers. We refer to these groups of data as *atomic* clusters.

For example, suppose one worker groups objects into a cluster of tall objects and

another of short objects, while a different worker groups the same objects into a cluster of red objects and another of blue objects. Then, our method should recover four atomic clusters: tall red objects, short red objects, tall blue objects, and short blue objects. The behavior of the two workers may then be summarized using a confusion table of the atomic clusters (see Section 5.4.3). The first worker groups the first and third atomic cluster into one category and the second and fourth atomic cluster into another category. The second worker groups the first and second atomic clusters into a category and the third and fourth atomic clusters into another category.

5.4.1 Generative Model

We propose an approach in which data items are represented as points in a Euclidean space and workers are modeled as pairwise binary classifiers in this space. Atomic clusters are then obtained by clustering these inferred points using a Dirichlet process mixture model, which estimates the number of clusters [Lo84]. The advantage of an intermediate Euclidean representation is that it provides a compact way to capture the characteristics of each data item. Certain items may be inherently more difficult to categorize, in which case they may lie between clusters. Items may be similar along one axis but different along another (e.g., object height versus object color.) A similar approach was proposed by Welinder et al. [WBBP10] for the analysis of classification labels obtained from crowdsourcing services. This method does not apply to our problem, since it involves binary labels applied to single data items rather than to pairs, and therefore requires that categories be defined a priori and agreed upon by all workers, which is incompatible with the crowdclustering problem.

We propose a probabilistic latent variable model that relates pairwise binary labels to hidden variables associated with both workers and images. The graphical model is shown in Figure 5.1. $\mathbf{x}_i$ is a D dimensional vector, with components $[\mathbf{x}_i]_d$ that encodes item i's location in the embedding space $\mathbb{R}^D$. Symmetric matrix $\mathbf{W}_j \in \mathbb{R}^{D \times D}$ with entries $[\mathbf{W}_j]_{d_1 d_2}$ and bias $\tau_j \in \mathbb{R}$ are used to define a pairwise binary classifier, explained in the next paragraph, that represents worker j's labeling behavior. Because

$\mathbf{W}_j$ is symmetric, we need only specify its upper triangular portion: vecp$\{\mathbf{W}_j\}$ which is a vector formed by "stacking" the partial columns of $\mathbf{W}_j$ according to the ordering $[\text{vecp}\{\mathbf{W}_j\}]_1 = [\mathbf{W}_j]_{11}, [\text{vecp}\{\mathbf{W}_j\}]_2 = [\mathbf{W}_j]_{12}, [\text{vecp}\{\mathbf{W}_j\}]_3 = [\mathbf{W}_j]_{22}$, etc. $\mathbf{\Phi}_k = \{\boldsymbol{\mu}_k, \mathbf{\Sigma}_k\}$ are the mean and covariance parameters associated with the k-th Gaussian atomic cluster, and U_k are stick breaking weights associated with a Dirichlet process.

The key term is the pairwise quadratic logistic regression likelihood that captures worker j's tendency to label the pair of images a_t and b_t with l_t:

$$p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) = \frac{1}{1 + \exp(-l_t A_t)} \tag{5.1}$$

where we define the pairwise quadratic *activity* $A_t = \mathbf{x}_{a_t}^T \mathbf{W}_{j_t} \mathbf{x}_{b_t} + \tau_{j_t}$. Symmetry of $\mathbf{W}_j$ ensures that $p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}) = p(l_t|\mathbf{x}_{b_t}, \mathbf{x}_{a_t}, \mathbf{W}_{j_t}, \tau_{j_t})$. This form of likelihood yields a compact and tractable method of representing classifiers defined over pairs of points in Euclidean space. Pairs of vectors with large pairwise activity tend to be classified as being in the same category, and in different categories otherwise. We find that this form of likelihood leads to tightly grouped clusters of points $\mathbf{x}_i$ that are then easily discovered by mixture model clustering.

The joint distribution is

$$\begin{aligned} p(\Phi, U, Z, X, W, \tau, \mathcal{L}) = &\prod_{k=1}^{\infty} p(U_k|\alpha) p(\Phi_k|\mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) \prod_{i=1}^{N} p(z_i|U) p(\mathbf{x}_i|\Phi_{z_i}) \\ &\prod_{j=1}^{J} p(\text{vecp}\{\mathbf{W}_j\}|\sigma_0^w) p(\tau_j|\sigma_0^\tau) \prod_{t=1}^{T} p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}). \end{aligned} \tag{5.2}$$

The conditional distributions are defined as follows:

$$
\begin{aligned}
p(U_k|\alpha) &= \text{Beta}(U_k; 1, \alpha) \\
p(z_i = k|U) &= U_k \prod_{l=1}^{k-1}(1 - U_l) \\
p(\mathbf{x}_i|\Phi_{z_i}) &= \text{Normal}(\mathbf{x}_i; \mu_{z_i}, \mathbf{\Sigma}_{z_i}) \\
p(\mathbf{x}_i|\sigma_0^x) &= \prod_d \text{Normal}([\mathbf{x}_i]_d; 0, \sigma_0^x) \\
p(\text{vecp}\{\mathbf{W}_j\}|\sigma_0^w) &= \prod_{d_1 \leq d_2} \text{Normal}([\mathbf{W}_j]_{d_1 d_2}; 0, \sigma_0^w) \\
p(\tau_j|\sigma_0^\tau) &= \text{Normal}(\tau_j; 0, \sigma_0^\tau) \\
p(\Phi_k|\mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) &= \text{Normal-Wishart}(\Phi_k; \mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0)
\end{aligned}
\tag{5.3}
$$

where $(\sigma_0^x, \sigma_0^\tau, \sigma_0^w, \alpha, \mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0)$ are fixed hyper-parameters. Our model is similar to that of [SST09], which is used to model binary relational data. Salient differences include our use of a logistic rather than a Gaussian likelihood, and our enforcement of the symmetry of $\mathbf{W}_j$. In the next section, we develop an efficient deterministic inference algorithm to accommodate much larger data sets than the sampling algorithm used in [SST09].

5.4.2 Approximate Inference

Exact posterior inference in this model is intractable, since computing it involves integrating over variables with complex dependencies. We therefore develop an inference algorithm based on the Variational Bayes method [Att99]. The high level idea is to work with a factorized proxy posterior distribution that does not model the full complexity of interactions between variables; it instead represents a single mode of the true posterior. Because this distribution is factorized, integrations involving it

become tractable. We define the proxy distribution $q(\Phi, U, Z, X, W, \tau) =$

$$\prod_{k=K+1}^{\infty} p(U_k|\alpha)p(\Phi_k|\mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) \prod_{k=1}^{K} q(U_k)q(\Phi_k) \prod_{i=1}^{N} q(z_i)q(\mathbf{x}_i) \prod_{j=1}^{J} q(\text{vecp}\{\mathbf{W}_j\})q(\tau_j) \tag{5.4}$$

using parametric distributions of the following form:

$$\begin{aligned} q(U_k) &= \text{Beta}(U_k; \xi_{k,1}, \xi_{k,2}) \\ q(\Phi_k) &= \text{Normal-Wishart}(\mathbf{m}_k, \beta_k, \mathbf{J}_k, \eta_k) \\ q(\mathbf{x}_i) &= \prod_d \text{Normal}([\mathbf{x}_i]_d; [\boldsymbol{\mu}_i^x]_d, [\boldsymbol{\sigma}_i^x]_d) \\ q(\tau_j) &= \text{Normal}(\tau_j; \mu_j^\tau, \sigma_j^\tau) \\ q(z_i = k) &= q_{ik} \\ q(\text{vecp}\{\mathbf{W}_j\}) &= \prod_{d_1 \le d_2} \text{Normal}([\mathbf{W}_j]_{d_1 d_2}; [\boldsymbol{\mu}_j^w]_{d_1 d_2}, [\boldsymbol{\sigma}_j^w]_{d_1 d_2}) \end{aligned} \tag{5.5}$$

To handle the infinite number of mixture components, we follow the approach of [KWV07] where we define variational distributions for the first K components, and fix the remainder to their corresponding priors. $\{\xi_{k,1}, \xi_{k,2}\}$ and $\{\mathbf{m}_k, \beta_k, \mathbf{J}_k, \eta_k\}$ are the variational parameters associated with the k-th mixture component. $q(z_i = k) = q_{ik}$ form the factorized assignment distribution for item i. $\boldsymbol{\mu}_i^x$ and $\boldsymbol{\sigma}_i^x$ are variational mean and variance parameters associated with data item i's embedding location. $\boldsymbol{\mu}_j^w$ and $\boldsymbol{\sigma}_j^w$ are symmetric matrix variational mean and variance parameters associated with worker j, and μ_j^τ and σ_j^τ are variational mean and variance parameters for the bias τ_j of worker j. We use diagonal covariance Normal distributions over $\mathbf{W}_j$ and $\mathbf{x}_i$ to reduce the number of parameters that must be estimated.

Next, we define a utility function which allows us to determine the variational parameters. We use Jensen's inequality to develop a lower bound to the log evidence:

$$\begin{aligned} &\log p(\mathcal{L}|\sigma_0^x, \sigma_0^\tau, \sigma_0^w, \alpha, \mathbf{m}_0, \beta_0, \mathbf{J}_0, \eta_0) \\ &\quad \ge E_q \log p(\Phi, U, Z, X, W, \tau, \mathcal{L}) + \mathcal{H}\{q(\Phi, U, Z, X, W, \tau)\}, \end{aligned} \tag{5.6}$$

$\mathcal{H}\{\cdot\}$ is the entropy of the proxy distribution, and the lower bound is known as the *Free Energy*. However, the Free Energy still involves intractable integration, because the normal distributions over variables $\mathbf{W}_j$, $\mathbf{x}_i$, and τ_j are not conjugate [BS94] to the logistic likelihood term. We therefore locally approximate the logistic likelihood with an unnormalized Gaussian function lower bound, which is the left hand side of the following inequality:

$$g(\Delta_t)\exp\{(l_t A_t - \Delta_t)/2 + \lambda(\Delta_t)(A_t^2 - \Delta_t^2)\} \leq p(l_t|\mathbf{x}_{a_t}, \mathbf{x}_{b_t}, \mathbf{W}_{j_t}, \tau_{j_t}). \tag{5.7}$$

This was adapted from [JJ96] to our case of quadratic pairwise logistic regression. Here $g(x) = (1 + e^{-x})^{-1}$ and $\lambda(\Delta) = [1/2 - g(\Delta)]/(2\Delta)$. This expression introduces an additional variational parameter Δ_t for each label, which are optimized in order to tighten the lower bound. Our utility function is therefore:

$$\begin{aligned}\mathcal{F} =& E_q \log p(\Phi, U, Z, X, W, \tau) + \mathcal{H}\{q(\Phi, U, Z, X, W, \tau)\} \\ &+ \sum_t \log g(\Delta_t) + \frac{l_t}{2} E_q\{A_t\} - \frac{\Delta_t}{2} + \lambda(\Delta_t)(E_q\{A_t^2\} - \Delta_t^2)\end{aligned} \tag{5.8}$$

which is a tractable lower bound to the log evidence. Optimization of variational parameters is carried out in a coordinate ascent procedure, which exactly maximizes each variational parameter in turn while holding all others fixed. This is guaranteed to converge to a local maximum of the utility function. The update equations are given in an extended technical report [GWKP11b]. We initialize the variational parameters by carrying out a layerwise procedure: first, we substitute a zero mean isotropic normal prior for the mixture model and perform variational updates over $\{\boldsymbol{\mu}_i^x, \boldsymbol{\sigma}_i^x, \boldsymbol{\mu}_j^w, \boldsymbol{\sigma}_j^w, \mu_j^\tau, \sigma_j^\tau\}$. Then we use $\boldsymbol{\mu}_i^x$ as point estimates for $\mathbf{x}_i$ and update $\{\mathbf{m}_k, \beta_k, \mathbf{J}_k, \eta_k, \xi_{k,1}, \xi_{k,2}\}$ and determine the initial number of clusters K as in [KWV07]. Finally, full joint inference updates are performed. Their computational complexity is $O(D^4T + D^2KN) = O(D^4NVRM + D^2KN)$.

5.4.3 Worker Confusion Analysis

As discussed in Section 5.4, we propose to understand a worker's behavior in terms of how he groups atomic clusters into his own notion of categories. We are interested in the predicted confusion matrix $\mathbf{C}_j$ for worker j, where

$$[\mathbf{C}_j]_{k_1k_2} = E_q\Big\{ \int p(l=1|\mathbf{x}_a, \mathbf{x}_b, \mathbf{W}_j, \tau_j) p(\mathbf{x}_a|\Phi_{k_1}) p(\mathbf{x}_b|\Phi_{k_2}) d\mathbf{x}_a d\mathbf{x}_b \Big\} \tag{5.9}$$

which expresses the probability that worker j assigns data items sampled from atomic cluster k_1 and k_2 to the same cluster, as predicted by the variational posterior. This integration is intractable. We use the expected values $E\{\Phi_{k_1}\} = \{\mathbf{m}_{k_1}, \mathbf{J}_{k_1}/\eta_{k_1}\}$ and $E\{\Phi_{k_2}\} = \{\mathbf{m}_{k_2}, \mathbf{J}_{k_2}/\eta_{k_2}\}$ as point estimates in place of the variational distributions over Φ_{k_1} and Φ_{k_2}. We then use Jensen's inequality and Eq. 5.7 again to yield a lower bound. Maximizing this bound over Δ yields

$$[\hat{\mathbf{C}}_j]_{k_1k_2} = g(\hat{\Delta}_{k_1k_2j}) \exp\{(\mathbf{m}_{k_1}^T \boldsymbol{\mu}_j^w \mathbf{m}_{k_2} + \mu_j^\tau - \hat{\Delta}_{k_1k_2j})/2\} \tag{5.10}$$

which we use as our approximate confusion matrix, where $\hat{\Delta}_{k_1k_2j}$ is given in [GWKP11b].

5.5 Experiments

We tested our method on four image data sets that have established "ground truth" categories, which were provided by a single human expert. These categories do not necessarily reflect the uniquely valid way to categorize the data set, however they form a convenient baseline for the purpose of quantitative comparison. We used 1000 images from the Scenes data set from [FFP05] to illustrate our approach (Figures 5.2, 5.3, and 5.4.) We used 1354 images of birds from 10 species in the CUB-200 data set [WBM$^+$10] (Table 5.1) and the 3845 images in the Stonefly9 data set [MMLM$^+$09] (Table 5.1) in order to compare our method quantitatively to other cluster aggregation methods. We used the 37794 images from the Attribute Discovery data set [BBS10] in order to demonstrate our method on a large scale problem.

We set the dimensionality of $\mathbf{x}_i$ to $D = 4$ (since higher dimensionality yielded no additional clusters) and we iterated the update equations 100 times, which was enough for convergence. Hyperparameters were tuned once on synthetic pairwise labels that simulated 100 data points drawn from 4 clusters, and fixed during all experiments.

Figure 5.2 (left) shows the mean locations of the data items $\boldsymbol{\mu}_i^x$ learned from the Scene data set, visualized as points in Euclidean space. We find well separated clusters whose labels k are displayed at their mean locations $\mathbf{m}_k$. The points are colored according to $\text{argmax}_k\, q_{ik}$, which is item i's MAP cluster assignment. The cluster labels are sorted according to the number of assigned items, with cluster 1 being the largest. The axes are the first two Fisher discriminant directions (derived from the MAP cluster assignments) as axes. The clusters are well separated in the four dimensionsal space (we give the average assignment entropy $-\frac{1}{N}\sum_{ik} q_{ik} \log q_{ik}$ in the figure title, which shows little cluster overlap.) Figure 5.2 (center) shows six high confidence examples from clusters 1 through 5. Figure 5.2 (right) shows the confusion table between the ground truth categories and the MAP clustering. We find that the MAP clusters often correspond to single ground truth categories, but they sometimes combine ground truth categories in reasonable ways. See Section 5.5.1 for a discussion and potential solution of this issue.

Figure 5.3 (left of line) shows the predicted confusion matrices (Section 5.4.3) associated with the 40 workers that performed the most HITs. This matrix captures the worker's tendency to label items from different atomic clusters as being in the same or different category. Figure 5.3 (right of line) shows in detail the predicted confusion matrices for three workers. We have sorted the MAP cluster indices to yield approximately block diagonal matrices, for ease of interpretation. Worker 9 makes relatively fine grained distinctions, including separating clusters 1 and 9 that correspond to the indoor categories and the bedroom scenes, respectively. Worker 45 combines clusters 5 and 8 which correspond to city street and highway scenes in addition to grouping together all indoor scene categories. The finer grained distinctions made by worker 9 may be a result of performing more HITs (74) and seeing a larger number of images

Figure 5.2: Scene Dataset. Left: Mean locations $\boldsymbol{\mu}_i^x$ projected onto first two Fisher discriminant vectors, along with cluster labels superimposed at cluster means $\mathbf{m}_k$. Data items are colored according to their MAP label $\text{argmax}_k\, q_{ik}$. Center: High confidence example images from the largest five clusters (rows correspond to clusters.) Right: Confusion table between ground truth scene categories and inferred clusters. The first cluster includes three indoor ground truth categories, the second includes *forest* and *open country* categories, and the third includes two urban categories. See Section 5.5.1 for a discussion and potential solution of this issue.

than worker 45, who performed 15 HITs. Finally (far right), we find a worker whose labels do not align with the atomic clustering. Inspection of his labels show that they were entered largely at random.

Figure 5.4 (top left) shows the number of HITs performed by each worker according to descending rank. Figure 5.4 (bottom left) is a Pareto curve that indicates the percentage of the HITs performed by the most active workers. The Pareto principle (i.e., the law of the vital few) [Par96] roughly holds: the top 20% most active workers perform nearly 80% of the work. We wish to understand the extent to which the most active workers contribute to the results. For the purpose of quantitative comparisons, we use Variation of Information (VI) [Mei03] to measure the discrepancy between the inferred MAP clustering and the ground truth categorization. VI is a metric with strong information theoretic justification that is defined between two partitions (clusterings) of a data set; smaller values indicate a closer match and a VI of 0 means that two clusterings are identical. In Figure 5.4 (center) we incrementally remove the most active (blue) and least active (red) workers. Removal of workers corresponds to moving from right to left on the x-axis, which indicates the number of HITs used to learn the model. The results show that removing the large number of workers

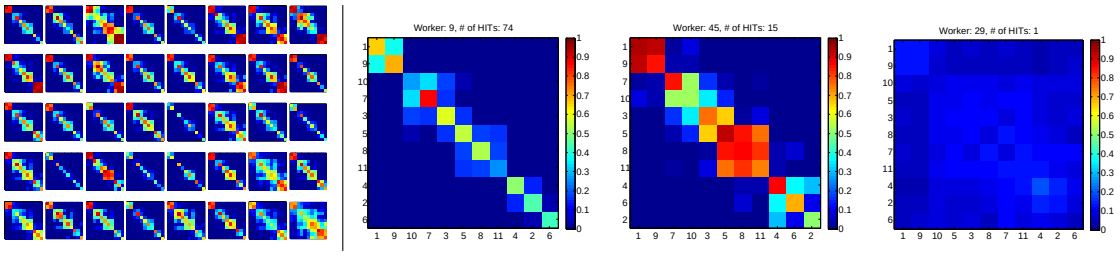

Figure 5.3: (Left of line) Worker confusion matrices for the 40 most active workers. (Right of line) Selected worker confusion matrices for Scenes experiment. Worker 9 (left) makes distinctions that correspond closely to the atomic clustering. Worker 45 (center) makes coarser distinctions, often combining atomic clusters. Right: Worker 29's single HIT was largely random and does not align with the atomic clusters.

that do fewer HITs is more detrimental to performance than removing the relatively few workers that do a large number of HITs (given the same number of total HITs), indicating that the atomic clustering is learned from the crowd at large.

In Figure 5.4 (right), we judge the impact of the sampling redundancy parameter V described in Section 5.3. We compare our approach (Bayesian crowdclustering) to two existing clustering aggregation methods from the literature: consensus clustering by nonnegative matrix factorization (NMF) [LDJ07] and the cluster ensembles method of Strehl and Ghosh (S&G) [SG02]. NMF and S&G require the number of inferred clusters to be provided as a parameter, and we set this to the number of ground truth categories. Even without the benefit of this additional information, our method (which automatically infers the number of clusters) outperforms the alternatives. To judge the benefit of modeling the characteristics of individual workers, we also compare against a variant of our model in which all HITs are treated as if they are performed by a single worker (Bayesian consensus.) We find a significant improvement. We fix $R = 5$ in this experiment, but we find a similar ranking of methods at other values of R. However, the performance benefit of the Bayesian methods over the existing methods increases with R.

We compare the four methods quantitatively on two additional data sets, with the results summarized in Table 5.1. In both cases, we instruct workers to categorize based on species. This is known to be a difficult task for non-experts. We set $V = 6$

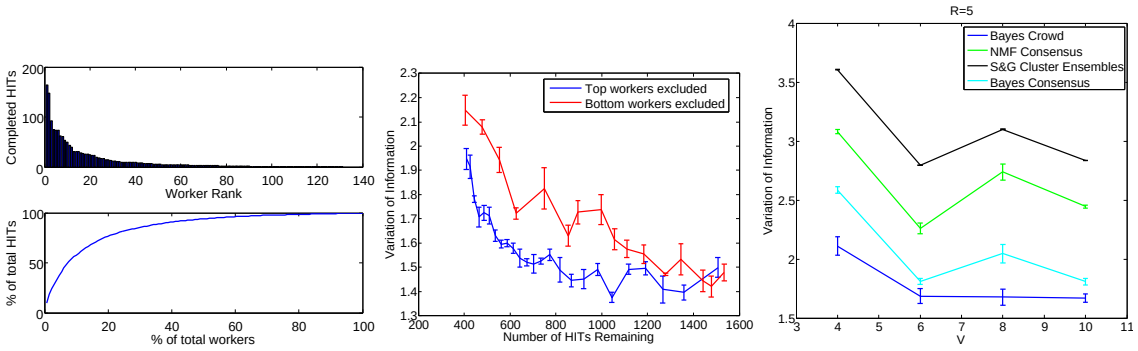

Figure 5.4: Scene Data set. Left Top: Number of completed HITs by worker rank. Left Bottom: Pareto curve. Center: Variation of Information on the Scene data set as we incrementally remove top (blue) and bottom (red) ranked workers. The top workers are removed one at a time, bottom ranked workers are removed in groups so that both curves cover roughly the same domain. The most active workers do not dominate the results. Right: Variation of Information between the inferred clustering and the ground truth categories on the Scene data set, as a function of sampling parameter V. R is fixed at 5.

	Bayes Crowd	Bayes Consensus	NMF [LDJ07]	Strehl & Ghosh [SG02]
Birds (VI)	$\mathbf{1.103 \pm 0.082}$	1.721 ± 0.07	1.500 ± 0.26	1.256 ± 0.001
Birds (time)	18.5 min	18.1 min	27.9 min	**0.93** min
Stonefly9 (VI)	$\mathbf{2.448 \pm 0.063}$	2.735 ± 0.037	4.571 ± 0.158	3.836 ± 0.002
Stonefly9 (time)	100.1 min	98.5 min	212.6 min	**46.5** min

Table 5.1: Quantitative comparison on Bird [WBM+10] and Stonefly [MMLM+09] species categorization data sets. Quality is measured using Variation of Information between the inferred clustering and ground truth. Bayesian Crowdclustering outperforms the alternatives.

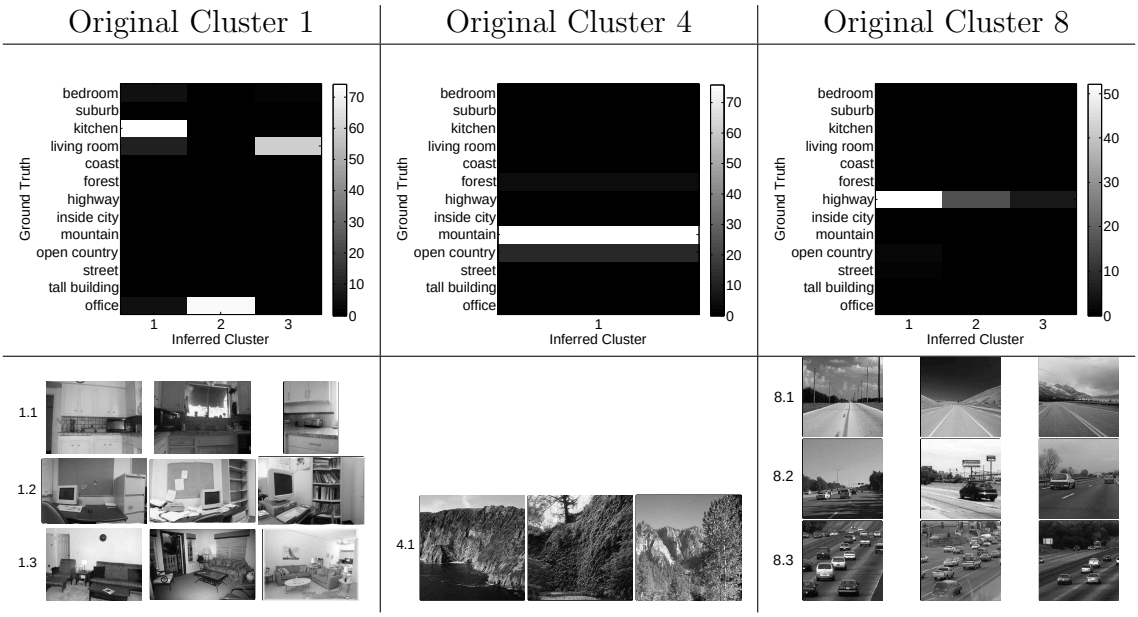

Figure 5.5: Divisive Clustering on the Scenes data set. Left: Confusion matrix and high confidence examples when running our method on images assigned to cluster one in the original experiment (Figure 5.2). The three indoor scene categories are correctly recovered. Center: Workers are unable to subdivide mountain scenes consistently and our method returns a single cluster. Right: Workers may find perceptually relevant distinctions not present in the ground truth categories. Here, the highway category is subdivided according to the number of cars present.

and $R = 5$ for these experiments. Again, we find that Bayesian Crowdclustering outperforms the alternatives. A run time comparison is also given in Table 5.1. Bayesian Crowdclustering results on the Bird and Stonefly data sets are summarized in [GWKP11b].

Finally, we demonstrate Bayesian crowdclustering on the large scale Attribute Discovery data set. This data set has four image categories: bags, earrings, ties, and women's shoes. In addition, each image is a member of one of 27 sub-categories (e.g., the bags category includes backpacks and totes as sub-categories.) See [GWKP11b] for summary figures. We find that our method easily discovers the four categories. The subcategories are not discovered, likely due to limited context associated with HITs with size $M = 36$ as discussed in the next section. Runtime was approximately 9.5 hours on a six core Intel Xeon machine.

5.5.1 Divisive Clustering

As indicated by the confusion matrix in Figure 5.2 (right), our method results in clusters that correspond to reasonable categories. However, it is clear that the data often has finer categorical distinctions that go undiscovered. We conjecture that this is a result of the limited context presented to the worker in each HIT. When shown a set of $M = 36$ images consisting mostly of different types of outdoor scenes and a few indoor scenes, it is reasonable for a worker to consider the indoor scenes as a unified category. However, if a HIT is composed purely of indoor scenes, a worker might draw finer distinctions between images of offices, kitchens, and living rooms. To test this conjecture, we developed a hierarchical procedure in which we run Bayesian crowdclustering independently on images that are MAP assigned to the same cluster in the original Scenes experiment.

Figure 5.5 (left) shows the results on the indoor scenes assigned to original cluster 1. We find that when restricted to indoor scenes, the workers do find the relevant distinctions and our algorithm accurately recovers the kitchen, living room, and office ground truth categories. In Figure 5.5 (center) we ran the procedure on images from original cluster 4, which is composed predominantly of mountain scenes. The algorithm discovers one subcluster. In Figure 5.5 (right) the workers divide a cluster into three subclusters that are perceptually relevant: they have organized them according to the number of cars present.

5.6 Conclusions

We have proposed a method for clustering a large set of data by distributing small tasks to a large group of workers. It is based on using a novel model of human clustering, as well as a novel machine learning method to aggregate worker annotations. Modeling both data item properties and the workers' annotation process and parameters appears to produce performance that is superior to existing clustering aggregation methods. Our study poses a number of interesting questions for further research: Can

adaptive sampling methods (as opposed to our random sampling) reduce the number of HITs that are necessary to achieve high quality clustering? Is it possible to model the workers' tendency to learn over time as they perform HITs, rather than treating HITs independently as we do here? Can we model contextual effects, perhaps by modeling the way that humans "regularize" their categorical decisions depending on the number and variety of items present in the task?

Chapter 6

Continuous Labels

6.1 Abstract

We explore crowdsourcing visual detection and localization tasks. Are all detection tasks similar? How many annotators are necessary to achieve good performance? Can one match the performance of an expert by combining the work of multiple annotators? We explored these questions by asking Amazon Mechanical Turk annotators to detect objects in approximately 1100 images from five different collections. The statistics of the five collections were chosen to explore potentially different conditions. Our experiments show that good precision may be achieved with few (2–3) annotators, while 10 annotators may be needed for high recall. Furthermore, expert performance may be approached by combining the detections of 10 annotators; however, it appears unlikely that it may be matched or surpassed even using a much larger number of naive annotators. Additional observations are that pay has little influence on the quality of the annotations, and that serializing the work of multiple annotators has little benefit over combining the work of annotators working in parallel.

6.2 Introduction

Crowdsourcing and citizen science are becoming increasingly important means for helping scientists collect and analyze large datasets of images. With the website

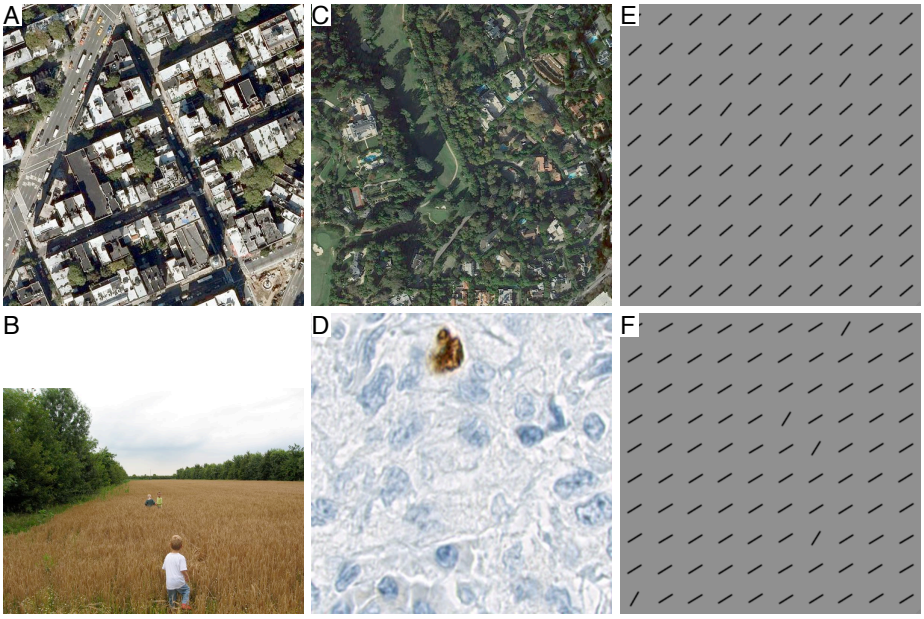

Figure 6.1: Examples images from the detection datasets. A. Yellow cabs (Section 6.3.2). B. People dataset (Section 6.3.3). C. Swimming pools (Section 6.3.1). D. Cell nuclei (Section 6.3.4). E–F. Difficult and easy task, respectively, from the diagonal bars dataset (Section 6.3.5).

Galaxy Zoo[1], astronomers enlist the help of amateurs to classify hundreds of thousands of photos of galaxies taken by the Hubble space telescope. The American space agency, NASA, asks citizen scientists to find craters on Mars by clicking on images on the planet surface through the "Be a Martian" website[2]. Closer to Earth, archaeologists crowdsourced the search for the tomb of Genghis Khan by asking ordinary people to look through satellite images of Mongolian plains[3]. Machine vision researchers have begun using crowdsourcing to clean up and annotate large datasets of images. This is especially useful to produce large and reliable training sets for visual recognition [RTMF08, SP08, DDS+09].

Recently there is much interest in crowdsourcing annotations in computer vision [SF08, DDS+09, RTMF08, YRLT09, WP10, WBBP10, VG09, VRP10] and other fields [SOJN08, SPI08, vAD04, vAMM+08]. For image annotations, previous work

[1] http://www.galaxyzoo.org
[2] http://beamartian.jpl.nasa.gov
[3] http://exploration.nationalgeographic.com

has focused on boundary tracing [SF08, RTMF08], part annotations [BM09, SF08], bounding boxes [WP10, VRP10], binary annotation [DS79, WRW+09, RYZ+09, WBBP10], and naming [SP08]. The accuracy of crowdsourcing has been explored for three: boundary tracing [SF08, RTMF08], naming [SP08] and binary labeling [WRW+09, WBBP10]. Visual detection and localization by crowdsourcing is a useful technique whose properties, especially in the case of multiple detections per image, have not yet been studied systematically.

Our goal here is to begin its exploration. The questions we are interested in are: (a) Are annotators accurate? (b) Are they similar in their performance? (c) Can one combine multiple 'naive' annotators and achieve the performance of an expert? (d) How much time and money does it take to annotate a large dataset? (e) What is the best way to organize the work of multiple annotators? (f) Are there qualitatively different annotation tasks? (g) What is the best way to aggregate the work of multiple annotators? We explore these questions by analyzing annotations obtained on a corpus of ∼1100 images belonging to five hand-picked image collections (Section 6.3). Our method is described in Section 6.4, the experimental results in Section 6.5. In Section 6.6 we study an iterative scheme for obtaining high quality detections. This experiment was inspired by TurKit [LCGM09], a toolbox for obtaining iterative annotations where the output of one annotator serves as the (modified) input to another annotator. We adopt here the terminology of [LCGM10] and consider both *iterative* and *parallel* annotation tasks. Our concluding remarks are collected in Section 6.7.

6.3 Datasets

We postulate that a number of different factors may cause errors (false alarms, missed targets) in detection and localization. First of all, if the dataset contains objects that look similar to the target, "red herrings" or distractor objects, it is likely that some of them will be mistakenly detected as object instances. Another case is when the objects themselves are sometimes difficult to detect due to weak "signal strength." This is often the case, for example, when target objects are very small in terms of pixel

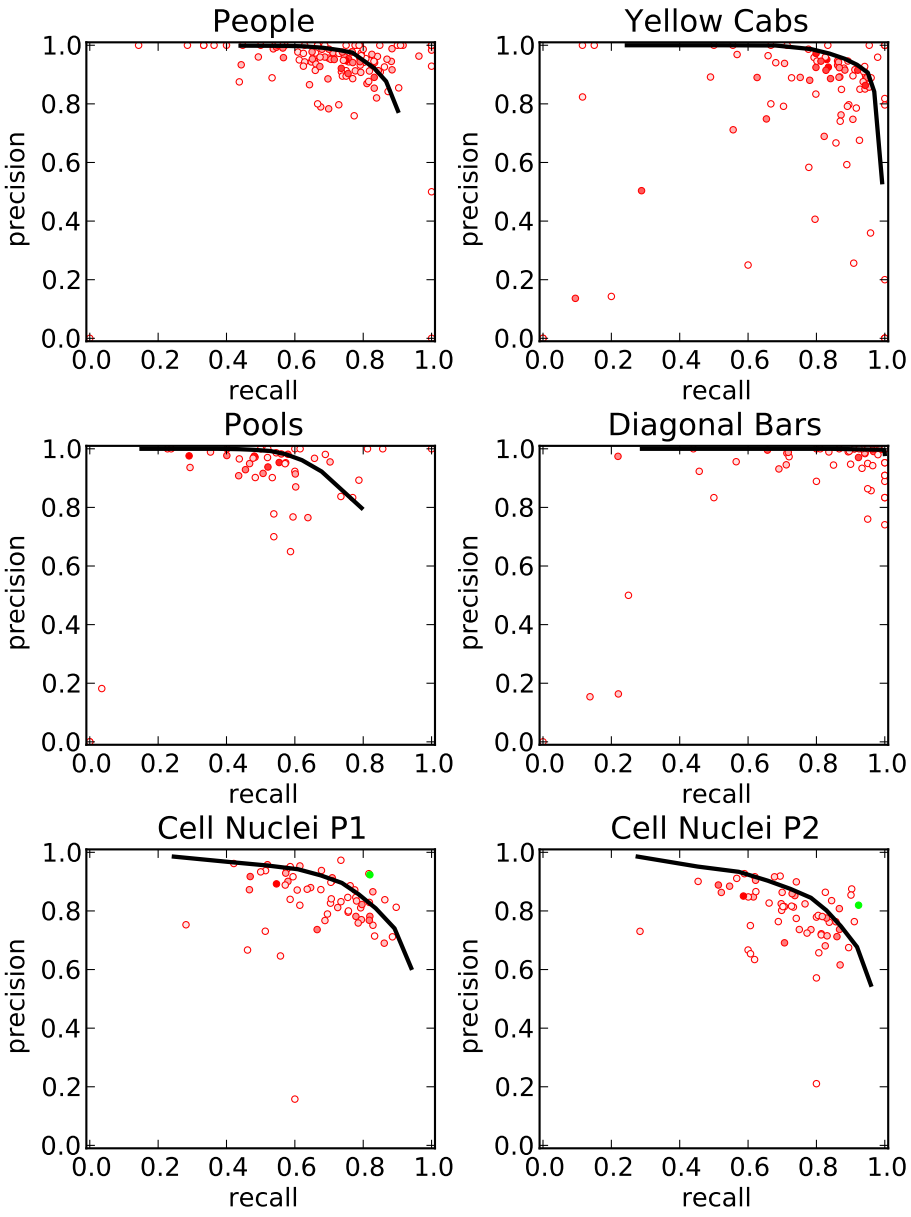

Figure 6.2: Annotator precision-recall. Each dot denotes one annotator. The curve is obtained by thresholding the number of clicks that each image location received. The bottom two plots show the precision vs. recall of the same annotations using the ground truth provided by two different experts. Each expert's precision-recall point vis-a-vis the other expert's ground truth is shown as a green dot.

size, or their contrast with respect to the background is faint. Other times it is tough to decide whether an object in the image is a single or dual instance of a target object due to "crowding." An example of this, if the target object category is people, would be one person almost completely occluding another one. Some annotators would see it as one person, while others would see it as two people very close together.

To highlight different issues in providing object detection annotations, we gathered five datasets. Each dataset exposes some of the issues discussed above to varying extents. The following sections describe each dataset in detail. With the exception of the last dataset, ground truth was established either by human operators who had access to higher resolution versions of the same images, or by human experts.

6.3.1 Swimming Pools

We collected 100 satellite images of a wealthy suburban neighborhood using Google Maps. The annotators were shown images of size 500×500 pixels with resolution of approximately 1 pixel per meter, and asked to look for swimming pools. The task is quite challenging as typical swimming pools are between 8 and 20 pixels wide and often lie in the shade of neighboring trees, thus their signal strength will be low. An initial version of the ground truth was provided by a hired annotator using annotation software developed in our group.

6.3.2 New York Yellow Cabs

We collected 240 satellite images of lower Manhattan from Google Maps. The images were sized at 500×500 pixels at a resolution of 2 pixels per meter. We asked the annotators to look for yellow cabs, which are clearly visible in many of the images. However, in some cases the cabs were driving in the shade of buildings, obscuring their distinctive yellow color (the signal strength). Similarly, they are sometimes hard to distinguish from white or yellowish non-cab cars (the red herrings). The hired annotator provided ground truth for this dataset on images at twice the resolution (4 pixels per meter).

6.3.3 People

The 521 images in this dataset originated from two sources. The vast majority were sampled from a collection of holiday images with mostly outdoor scenes. The remaining images were street scenes taken by an mobile phone at red light intersections. Images were either in portrait or landscape mode and the largest dimension was 800 pixels to ensure that they could be viewed in a standard web browser. 30 of the images contained no people at all. Since we consider only two-dimensional spatial detections, we asked annotators to click on the centroid of the head of any person found in the scene. We also explicitly asked them to ignore red herrings like statues, people-like toys and photos or posters of people in the images. The task is often quite easy, but can be challenging in crowded scenes or images where the people are very small in pixel-size. However, once a person has been found, it is usually pretty clear that it is a person, since people are not easily confused with other objects.

The ground truth was obtained by merging all detection annotations obtained from Amazon Mechanical Turk (MTurk) using a spatial clustering algorithm (see Section 6.4). One of the authors verified each of the consolidated annotations by overlaying such annotations on images that had 9–16x the resolution of the images that had been sent to the MTurk workers. Missed persons were annotated using the same technique.

6.3.4 Cell Nuclei

Patches of 320×320 were cropped from 1500×1500 pixel images of tissue samples [FHW+09] and up-sampled (using cubic interpolation) to 640×640 pixels, obtaining 136 images in total. Annotators were asked to click on cell nuclei according to some example images and written instructions. Cell nuclei were described as: *Shapes with a clearly defined boundary. They are often (but not always) round in shape. Sometimes only the blue boundary (membrane) of a nucleus can be seen, while the nucleus itself appears white. Nuclei can also appear as brown shapes (stained with a marker).* The task is challenging for several reasons. First, for most annotators this is the first time

they are exposed to images of tissue samples and cell nuclei annotation. Sometimes it is hard to know whether an image shows two neighboring nuclei or one single elongated nucleus. Some nuclei are strongly ambiguous and can be mistaken for non-nuclei tissue.

Two expert pathologists independently provided the ground truth detection annotations for the dataset. The pathologists had access to the full size tissue samples and used a UI where they could zoom and pan the image. Their task was to provide circular annotations around each nucleus, thus giving the scale in addition to the location of each nucleus. As ground truth, we used only the center of the circular annotation, and discarded the scale information. We kept the annotations from both annotators separate, and used one as the ground truth during the experiments, except where stated otherwise.

6.3.5 Diagonal Bars

We generated a more controlled dataset with images containing a 10×10 grid of bars at different orientations. Most bars were oriented at 45°, but in each image N bars were oriented at θ degrees from the diagonal. The annotators were asked to find bars that were *not* at 45° diagonal. We varied $N = \{5, 10, 20\}$ and $\theta = \{10°, 15°, 30°\}$ and generated 10 random images for each combination, yielding 90 images in total. All images were 800×800 pixels. Since the images were generated synthetically, we had access to the ground truth.

6.4 Method

For each of the datasets listed in Section 6.3, we asked annotators to click with their mouse on the centroid of each detection. Although the approach described in this paper could in principle be generalized to other kinds of detection primitives, such as bounding boxes or rotatable ellipses, we did not compare these methods in order to focus on the main question of interest in this study: detection and localization.

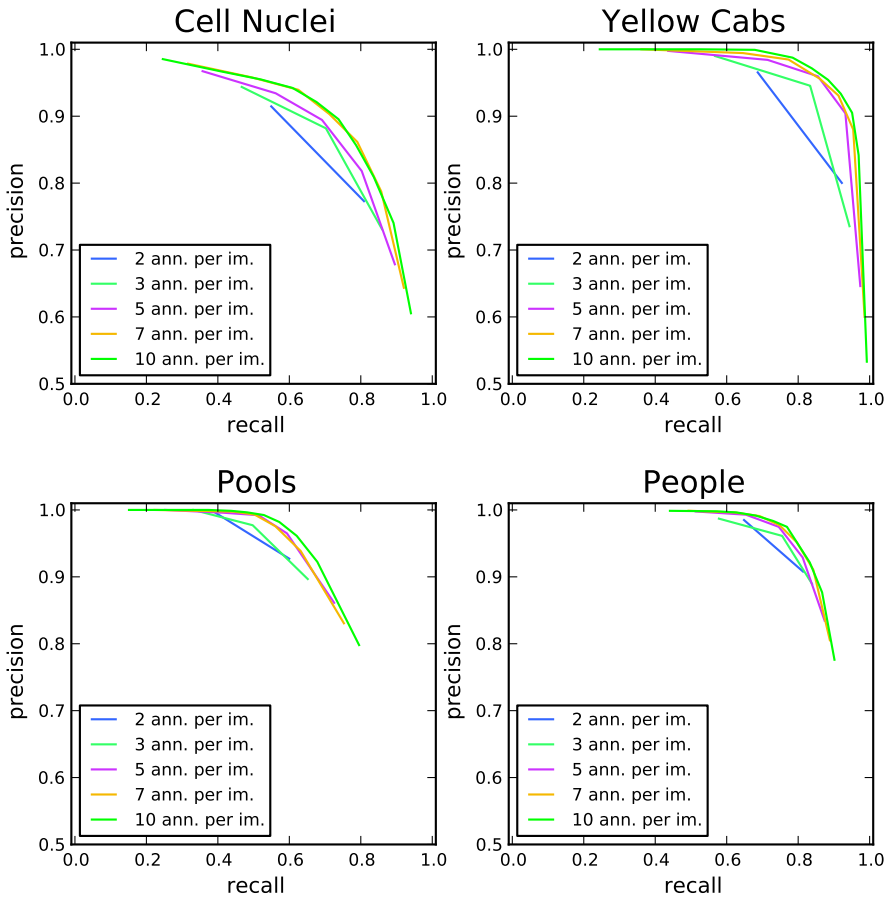

Figure 6.3: Precision-recall curves for different numbers of annotators.

To match a list of detections $X = \{x_1, x_2, \ldots, x_s\}$ against a ground truth annotation list $Y = \{y_1, y_2, \ldots, y_t\}$ we used a simple thresholding heuristic as follows. Let M be a list of matched pairs (x_i, y_j) that is initially empty, and let τ be some distance threshold. For each detection $x_i \in X$, the closest ground truth detection $y_j \in Y$ is found such that the distance $d(x_i, y_j) < \tau$ and y_j does not belong to a pair in M. If such a y_j is found, a reverse process takes place where we check if there is a detection x_k not in a pair in M that is closer to y_j. If a closer detection is found, (x_k, y_j) is added to M, otherwise (x_i, y_j) is added to M. The process continues until we have iterated through all $x_i \in X$ that are not in M. By comparing the sizes of X, Y and M we can compute the number of correct detections, the number of false alarms and the number misses in an image and, thus, precision and recall. In our experiments, we chose τ to be slightly larger than the largest object size for each dataset, which corresponds to $\tau = 10$ pixels for the yellow cabs, $\tau = 13$ pixels for the swimming pools, $\tau = 15$ pixels for the diagonal bars, $\tau = 20$ for the people dataset, and $\tau = 25$ pixels for the cell nuclei images.

Because multiple annotators annotated each image we could obtain a list of detections with a "confidence score" for each image location by clustering the detections from the different annotators and counting the number of members in each cluster. We used a basic clustering heuristic based on the matching algorithm described in the previous paragraph to obtain the cluster centers. The clustering algorithm is described in detail in the supplementary material.

6.5 Experiments

For each of the five datasets, we asked annotators on MTurk to provide detection annotations for the images. We paid 5 cents per image unless stated otherwise, and allowed up to 20 minutes for the annotator to complete the annotation of one image. Each image was annotated by 10 annotators. For each dataset, we provided written and pictorial instructions of what the annotators should be looking for. All tasks included 2–5 example images where the annotator was shown an image annotated

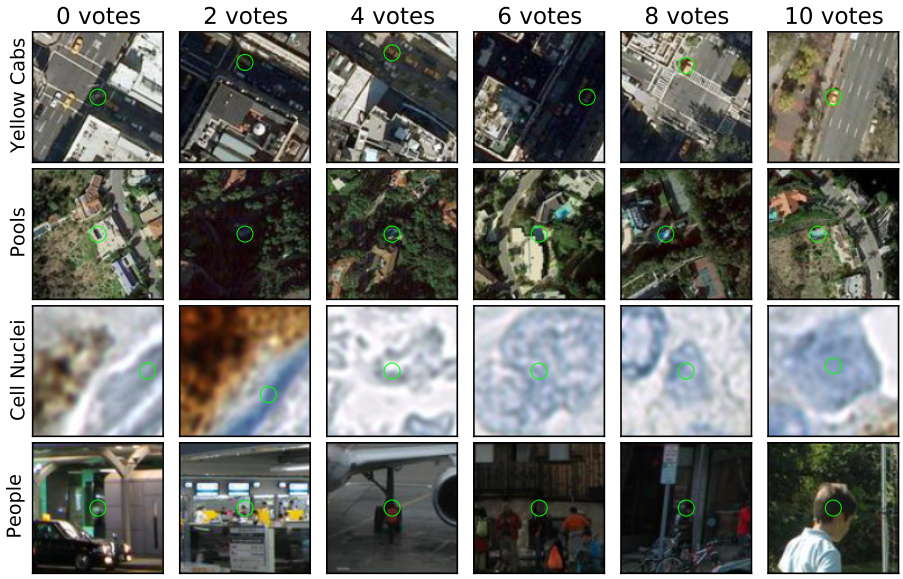

Figure 6.4: Examples of objects of different difficulty. Each image shows an object in the ground truth (marked by a green circle). The columns indicate how many of the annotators (out of 10) that detected the object, that is the number of "votes." No annotators detected the images in the left-most column.

with ground truth. For some of the datasets, like the cell nuclei, there was some ambiguity in how sensitive the annotators should be in providing annotations. To provide some guidance, and to allow them to choose freely for themselves, we gave all annotators the following instructions:

> You will often be uncertain as to whether to click somewhere or not. That is OK. Just follow your intuition and mark the spot (or leave it unmarked) without regrets. Your motto should be "I do the best I can." We will rely on the clicks from many people to come up with an ultimate score.

When working with annotators on MTurk, there is always the risk that some of the annotators will be "spammers." That is, they will do many tasks very quickly to make as much money as possible, in the hope that their sloppy work will go unnoticed. We encountered very few spammers (between 1–3% of the total number of annotators). Because they were so few, we did not exclude their data from our analysis.

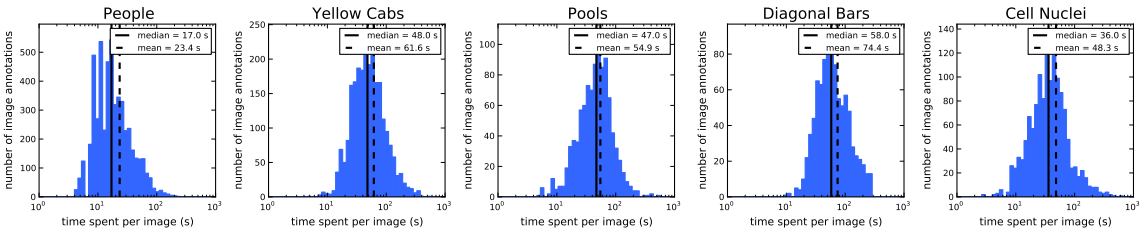

Figure 6.5: Distribution of the duration of time the annotators spent annotating each image (see Section 6.5.2).

6.5.1 Annotator Performance

The standard measures of performance in detection tasks are precision and recall. For a given detector, the *precision* is defined as the fraction of its detections that are correct according to a ground truth. The *recall* is the fraction of the total number of targets that have been detected by the detector. Treating the annotators as detectors, we can plot one of them as a point on the precision-recall plane, see Figure 6.2.

From Figure 6.2 we can see that there is usually a great spread in the precision-recall of the different annotators. This is due to three factors. First, each annotator annotates different images whose statistics may be somewhat different. A second reason is that many annotators only annotate few images, therefore our measurement of their precision and recall may be noisy. A third and last reason is that annotators may have different ability and may be using different criteria for detection [WBBP10]. As mentioned previously, some annotators are "spammers" who aim to do the task quickly but poorly.

On each plot in Figure 6.2 we show a precision-recall curve based on the votes per detection, as described in Section 6.4. The curves have ten steps since a detection can at most have ten votes from the different annotators. For the people and pools datasets, the point of highest recall on the curve is still 10–20% away from total recall. As mentioned in Section 6.3, this is because some swimming pools and people are hard to detect even at the resolution at which the ground truth was provided, not to mention the lower resolution the MTurk annotation tasks were carried out at. In contrast, in the diagonal bars experiment all target bars had at least one vote

(although there were also a small number of false alarms).

For the cell nuclei dataset we have two ground truth annotations from two different pathologists, and so we show a separate precision-recall plot for each one. On each plot the other pathologist is shown as a green point. Qualitatively, the two plots look very similar. Some annotators actually show up as having better precision-recall than the experts. However, this is due to the annotators labeling only a few images and serendipitously getting most of the detections of those correct. Annotators that annotated more than 10 images perform worse than the experts. The precision-recall curve from the integrated MTurk annotators is also lower than the expert. On the other hand, the pathologists have years of training while the MTurk annotators only looked briefly at five training images.

Figure 6.3 shows precision-recall curves for a subsampled portion of each dataset. For each dataset, the curves converge as the number of annotators per image approaches 10. One interesting thing to note here is that if a detection has two or more votes, the precision is close to or above 90% in all datasets but the cell nuclei dataset. This means that in general, if an object was detected by two separate annotators, we can be pretty sure it is correct. The cell nuclei dataset is particularly challenging, as some things that look like nuclei to amateurs can be explained away as something else by an expert pathologist. Hence the lower precision in the cell nuclei dataset.

To get a sense of what causes the variation in the precision-recall plots, we show examples of object instances of varying difficulty from all datasets in Figure 6.4. As can be seen from the examples, the mistakes are often quite understandable. Moreover, as expected, clear object instances have a lot of detections. Figure 6.6 shows some examples of false alarms with at least two detections. With a few exceptions, most false alarms are detections of object instances that share some visual characteristics with the target object type.

Figure 6.6: Examples of some typical false alarms (marked with green circles) that had at least two votes.

6.5.2 Time Spent on Tasks

Figure 6.5 shows how a histogram of the duration of time the annotators spent annotating each image. The variance in the duration is large for all datasets, and it can vary almost two orders of magnitude within the same dataset. However, since we require that each annotator reads the instructions at least once for each dataset, some of the outliers may be due to people spending a long time on the instructions. We use the median duration to compare the different datasets since it is more robust to outliers than the mean. We observe that there is quite a large difference in how long annotators spend on the datasets. While the median time to annotate an image from the people dataset is 17 seconds, it takes annotators almost a minute to label images from the diagonal bar dataset.

One reason that some datasets take longer to annotate is due to the statistics of the number of objects in images from the dataset. In Figure 6.7 we see that the time it takes to annotate an image is directly proportional to the number of detections an annotator provides for the image. Across the datasets, it takes between 1.3–2.8 seconds to label an object instance in an image. Furthermore there is also a "setup" cost associated with each dataset, characterized by the intercept of the fitted linear model. The entry cost varies substantially between the datasets, from less than 20 seconds for the people datasets up to almost a minute in the yellow cabs and the diagonal bar datasets.

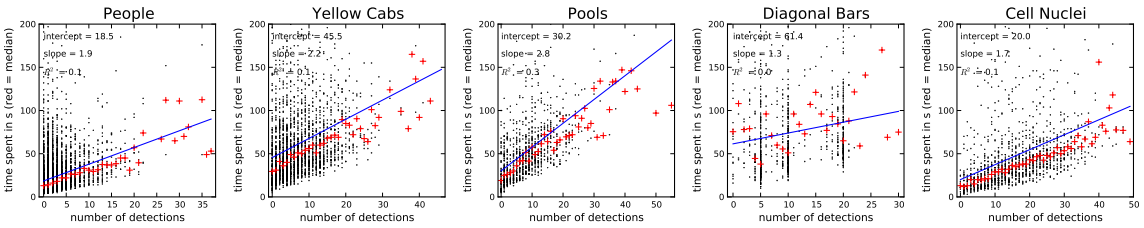

Figure 6.7: The time spent per image versus the number of detections the annotator reported for the image (see Section 6.5.2). The blue line is a least squares fitted linear model, the black dots are individual image annotation assignments, and the red crosses show median time spent for a given number of detections.

6.5.3 Influence of Pay on Time and Quality

When using a paid crowdsourcing service such as MTurk, there is always the question of whether compensation affects performance: Will the quality of the annotations go down if I pay less? Will the annotators be more careful if I pay more? We set out to investigate these questions by asking for annotations of the same dataset several times, but at different pay rates.

We chose the cell nuclei dataset as it is quite challenging, and annotators may be tempted to cheat if they are not paid enough. The dataset was sent to MTurk for annotation in three rounds with different pay rates per image: \$0.03, \$0.10 and \$0.20. We let some hours pass between each round, and we ensured that the same worker could label each image only once. In each of the three rounds we asked three separate annotators to annotate the same image.

Interestingly, as shown in Figure 6.8, the spread of annotator precision-recall appears to be very similar across the different pay rates. However, the precision-recall curve based on voting is slightly better for the \$0.20 than the lower rates. We also found that the time spent per image by annotators stays approximately constant but is slightly higher for the well-paid annotators (the medians for 3, 10 and 20 cents are 38, 36, and 41 seconds, respectively, see the supplementary materials). This suggests that there is only a slight, if any, influence of pay on time and quality. However, we did notice it took longer for low-paying jobs to be completed, suggesting that annotators are reluctant to start working on them.

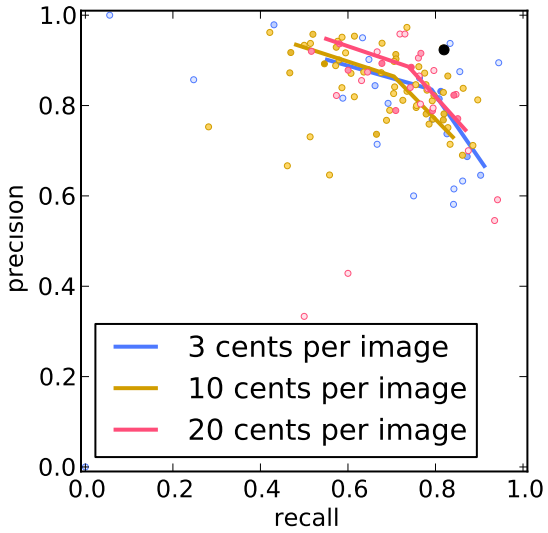

Figure 6.8: The precision-recall plane for the cell nuclei dataset for different pay rates. Each marker is a different annotator. The black marker is the expert annotator. The curves are the precision-recall curves obtained from clustering and voting (see Section 6.5.3).

6.6 Iterative Detection

This section compares the parallel annotation process used in Section 6.5 with an iterative scheme where annotators build on each others' annotations. This is similar to the work of Little et al. on iterative human computation [LCGM09, LCGM10] which shows that humans working in an iterative manner, improving on each others results, can perform better than humans working in parallel on certain tasks.

We constructed a task on MTurk where annotators were asked to provide detection annotations for the different datasets, just as in Section 6.5 with one significant difference: locations in the images that had previously been clicked twice were marked by a red circle. The annotators were told that symbols indicated detections that had been carried out by other annotators, but that they should look for and provide detection annotations for any objects instances that might have been missed by previous workers (see the supplementary materials for screenshots of the UI).

To initialize the overlaid red circles, we randomly sampled two annotators from the

experiments in Section 6.5.1 and clustered their detections (similar to [LCGM09] we call these "seed" annotations). We kept all clusters with two votes and overlaid them as circles on the image in the iterative MTurk annotation task. The performance of the sampled annotators can be seen as the red curves in the upper plots of Figure 6.9.

The task continued in an iterative manner: (1) The annotations (obtained from one annotator per image) were retrieved from MTurk. (2) The retrieved annotations were added to the seed annotations and clustered, and all clusters with two or more votes were overlaid on a new task that was sent to MTurk. The process iterated through steps (1) and (2).

The hypothesis was that (a) the iterative tasks would be quicker than the parallel ones as there are fewer detections to be made (see Figure 6.7), and that (b) the recall would increase as missed object instances were found by annotators not having to attend to already detected instances.

Figure 6.9 shows that the recall does increase with the iteration number for most of the datasets, with the precision being slightly impacted (due to more false alarms). Furthermore, the median time spent per task is generally 20–50% lower for the iterative tasks as compared to the parallel tasks in Figure 6.5, and stays approximately constant with iteration number. However, comparing Figure 6.9 with the curves for parallel tasks in Figure 6.3, it clear that parallel tasks perform better than iterative annotation.

6.7 Discussion and Conclusion

We carried out our investigation by asking MTurk workers to detect objects in 1100 images coming from five different collections. The datasets were chosen so as to probe visual detection on images with different statistics.

Our main observations are: (a) Annotation takes 20–50 seconds per image plus 1–3 seconds per detection, i.e. approximately one minute for the typical image in our database. Annotation times vary considerably across images and annotators. (b) Compensation of around 0.1–0.2$ / image is adequate for achieving the best results.

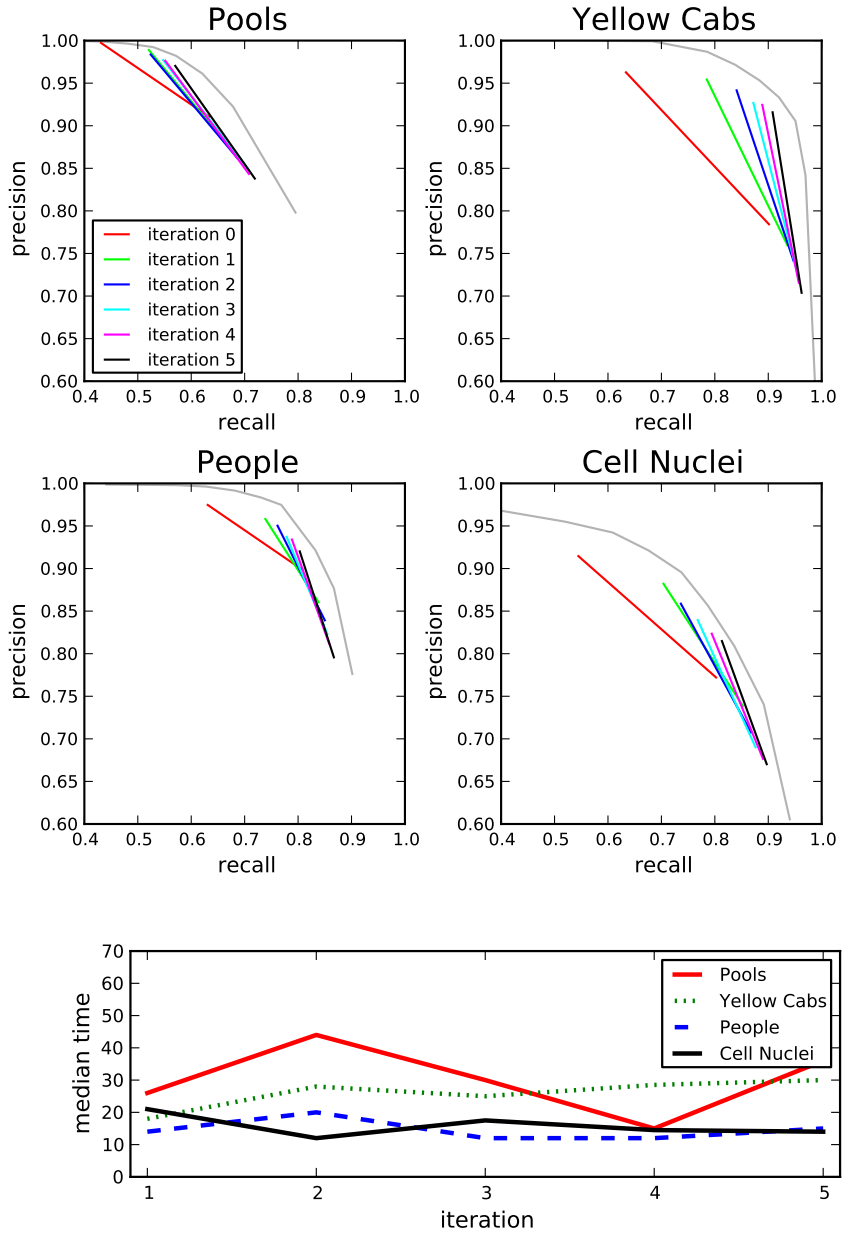

Figure 6.9: The iterative labeling task. Top: the improvement in the precision-recall with each iteration. Bottom: the median time (in seconds) annotator spent per image in each iteration for the different datasets.

This translates to approximately 5–10$/hour per annotator. (c) The precision-recall characteristics of different annotators varies considerably. However, annotators tend to group around the same precision-recall curve, indicating that their performance is comparable and what changes is how conservative they are in committing to a detection. The only annotators whose performance is significantly superior are those who, by annotating very few images, enjoy a 'lucky streak'. (d) The joint performance of 10 annotators approaches that of experts. However, it is unlikely that by combining very large numbers of naive annotators one could match or beat the performance of experts. (e) It is possible to achieve good precision with just two annotators, by selecting locations that are clicked by both. However, achieving high recall rates requires on the order of 10 annotators. (f) We do not find that serializing the work of multiple annotators has significant advantages with respect to combining the parallel work of the same number of annotators.

In the present study we combined the work of multiple annotators by voting. While this technique appears to be both robust and effective, it is possible that more sophisticated probabilistic methods might be more information-efficient by estimating the individual characteristics of the annotators as well as the difficulty of each detection [DS79, WRW+09, WBBP10].

6.8 Appendix: Annotation Statistics

6.8.1 Annotator Activity

How much does each annotator annotate? The histograms in Figure 6.10 show the distribution of the annotators over the size of the jobs they carried out. Notice that log scale was used for the horizontal axis. The histograms show that annotators are, to a first order of approximation, distributed uniformly over the log scale. The right skew of the 'People' plot may indicate that this was an enjoyable task and that annotators decided to annotate more images, while the left skew of the 'Diagonal Bars' plot may mean that the task was less enjoyable.

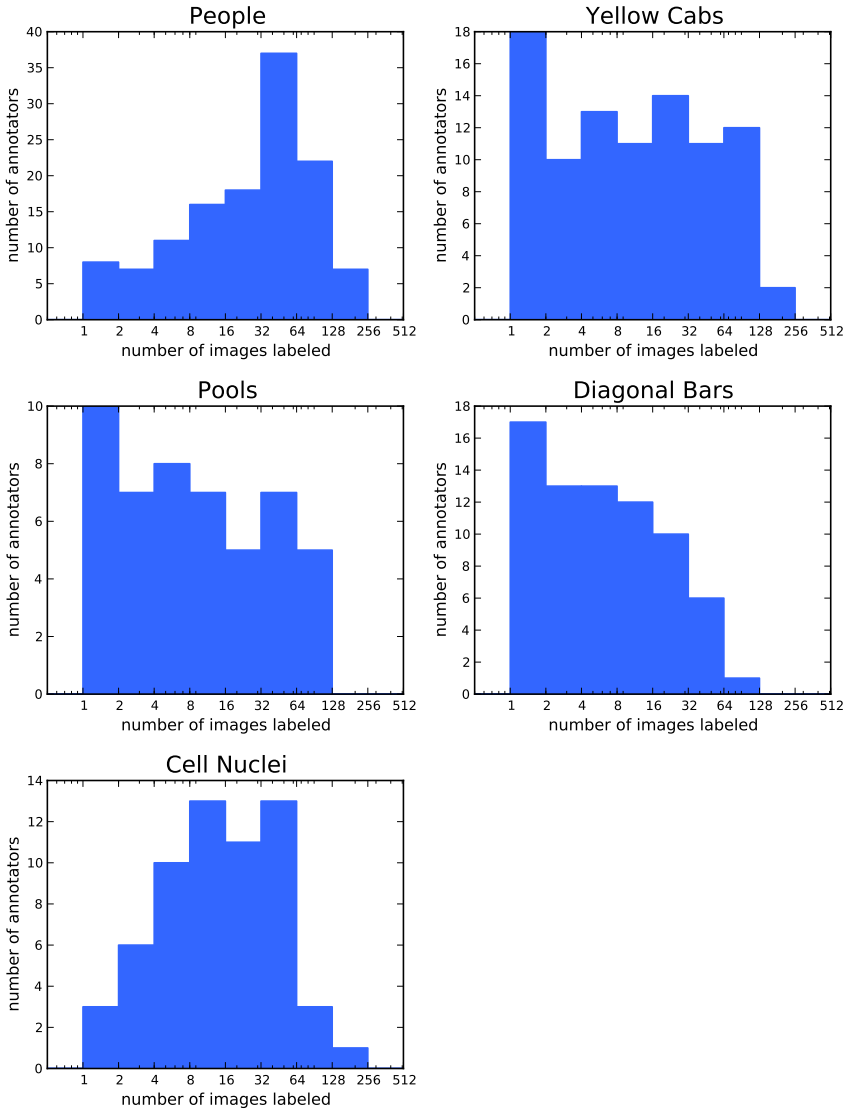

Figure 6.10: Annotator activity. Histograms showing the distribution of annotator activity, as measured by how many images the annotators annotated.

6.8.2 Error versus Annotator Activity

Do annotators that annotate a lot of images do a good job on those images? In Figure 6.11, each axes shows the total error rate of the annotators versus the number of images that the annotators annotated. The total error rate r is computed by $r = (f + m)/(m + c)$, where f is the number of false alarms, m is the number of misses, and c is the number of correct detections. An annotator is represented as a marker on the plot. Red markers indicate that the annotator annotated fewer than 10 images, black markers that the annotator annotated at least 10 images. The plots have been divided into three horizontal bands by hand, to highlight the "bad" (red), "good" (green), and "lucky" annotators (yellow). The bad annotators have high errors, the good annotators have low errors, even after annotating many images, and the lucky annotators manage to get very low error rates only because they annotate a few easy images.

6.8.3 Targets per Image

How many target objects are present in each image? The histograms in Figure 6.12 show the distribution of targets per image for each dataset. In the diagonal bars dataset we placed either 5, 10 or 20 target objects per image.

6.8.4 Task Acceptance Time

How long does it take for MTurk workers to start working on the task after it is published? Figure 6.13 show how quickly jobs are completed on MTurk.

Figure 6.14 shows same data as for Figure 6.13, but shown as the percentage of the total number of HITs completed at a given point in time. Note that the horizontal axes have different scale.

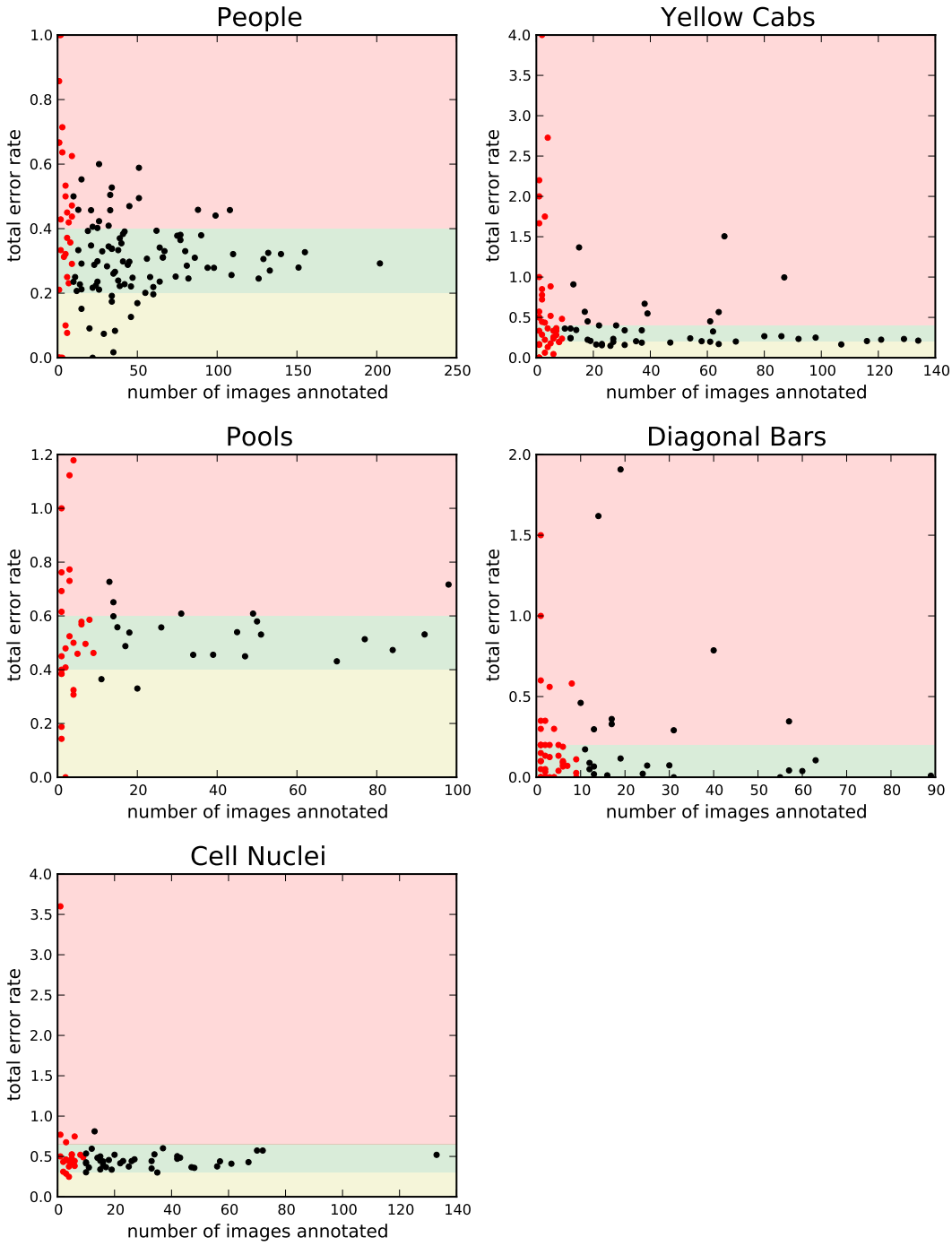

Figure 6.11: Annotator error rate versus activity. In each plot, an annotator is represented by a dot denoting the number of images he annotated, and the average error in the annotations. Red markers indicate that the annotator annotated fewer than 10 images, black markers that the annotator annotated at least 10 images. The plots have been divided into three horizontal bands by hand, to highlight the "bad" (red), "good" (green), and "lucky" annotators (yellow).

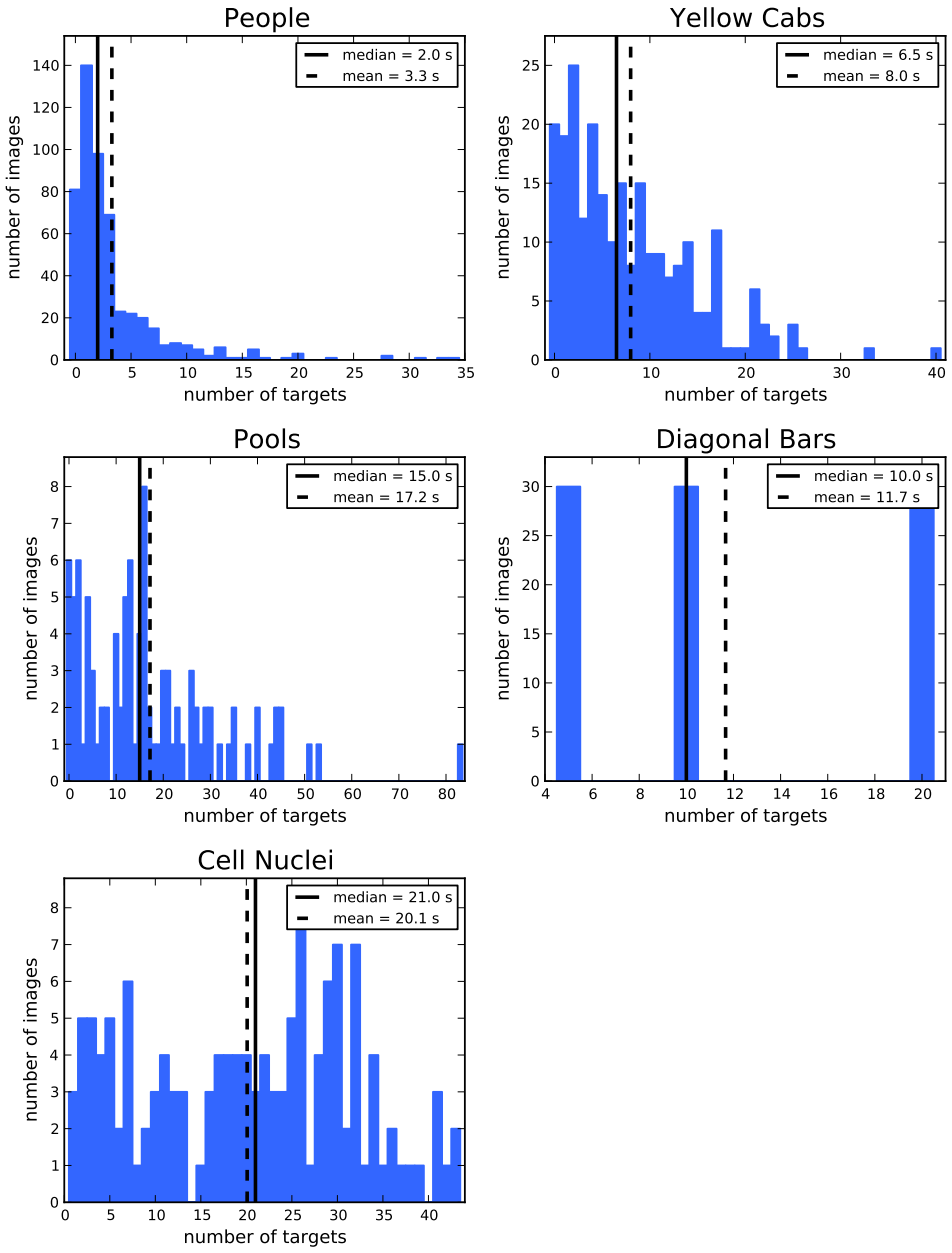

Figure 6.12: The density of targets per image. Each histogram shows the distribution of targets per image.

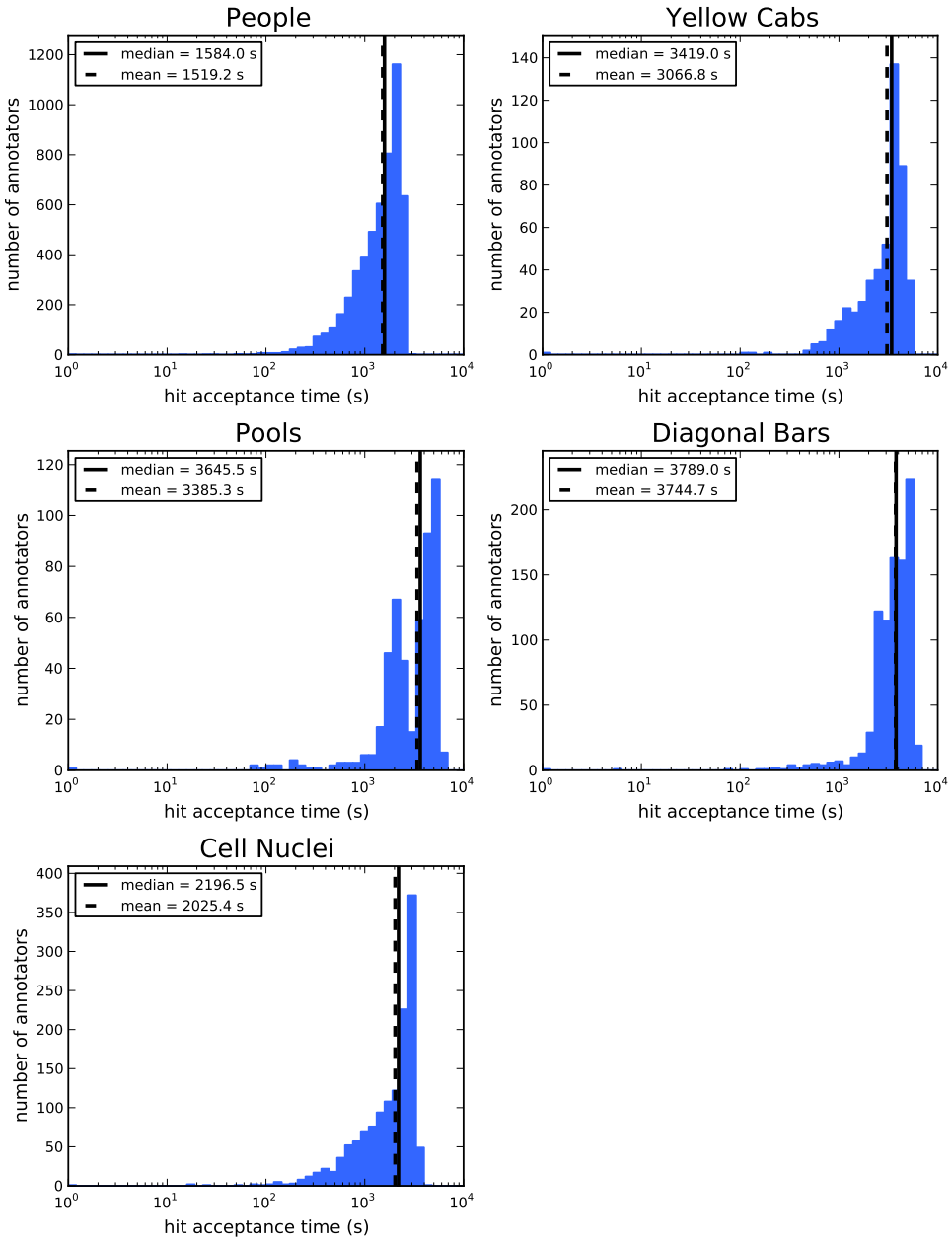

Figure 6.13: Time to completion for tasks on MTurk. The horizontal axis shows the time (in seconds) from the time the HITs were released on MTurk until a worker starts working on the HIT.

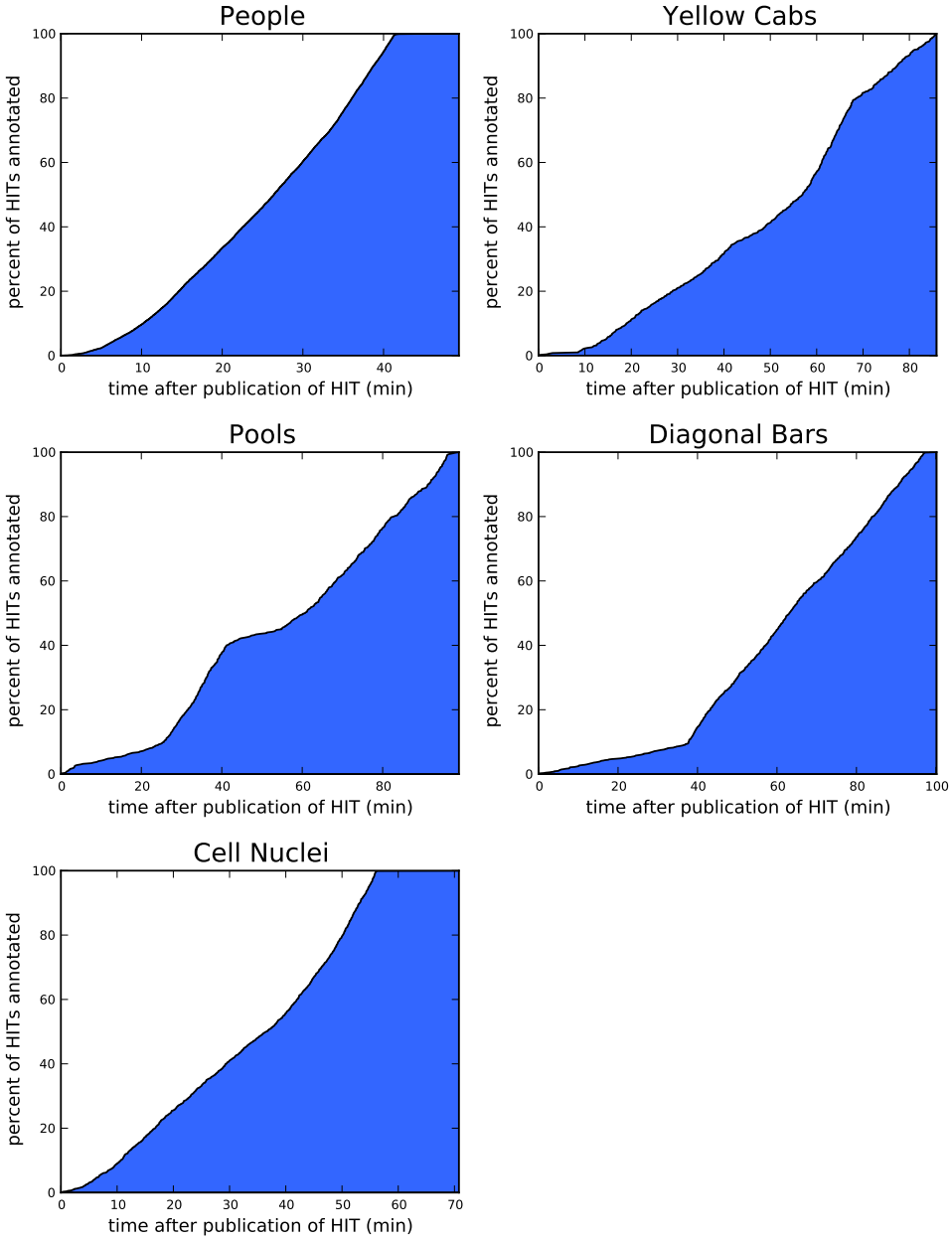

Figure 6.14: Percentage of HITs completed over time. Same data as in Figure 6.13 but plotted differently.

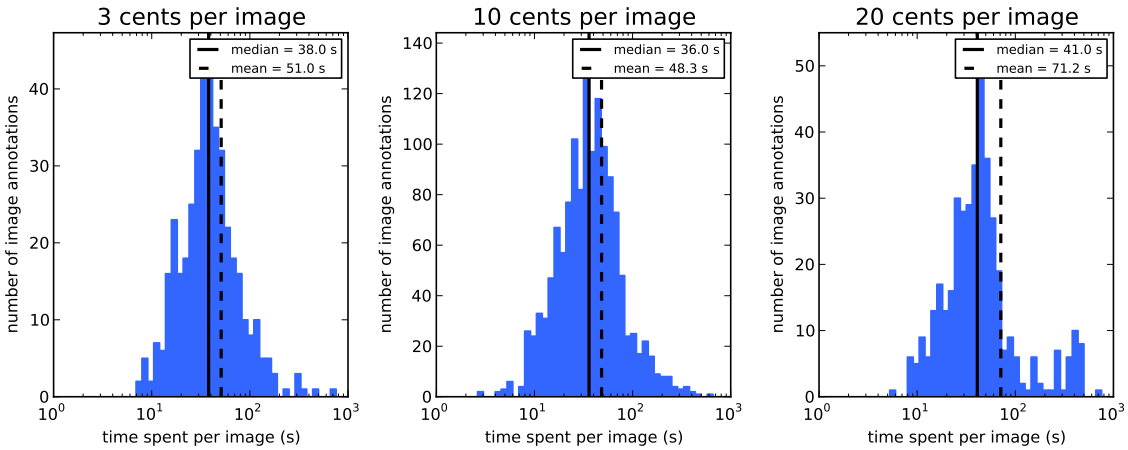

Figure 6.15: Time spent annotating images versus for three different levels of payment.

6.8.5 Time Spent For Different Levels of Payment

As references in the paper, the histograms in Figure 6.15 show how the distributions of time spent per task vary for different levels of payment ($0.03, $0.10, $0.20 per image annotated) on the cell nuclei dataset. There is no significant difference between the distributions.

6.8.6 Time Spent versus Number of Targets

Figure 6.16 shows the time spent per task versus the number of ground truth target objects in the image. Red marker is the median for a given number of targets. This is the similar to Figure 7 in the paper, but shows the number of *targets* instead of the number of *detections*.

6.8.7 Error versus Time Spent on Task

Do annotators that spend more time on a task have lower error rates? The plots in Figure 6.17 show the total error rate (as defined previously) versus the median time an annotator spent annotating an image. There seems to be little evidence that annotators that spend more time are better at the task. The annotators that annotated fewer than 10 images are shown as red dots while annotators that annotated

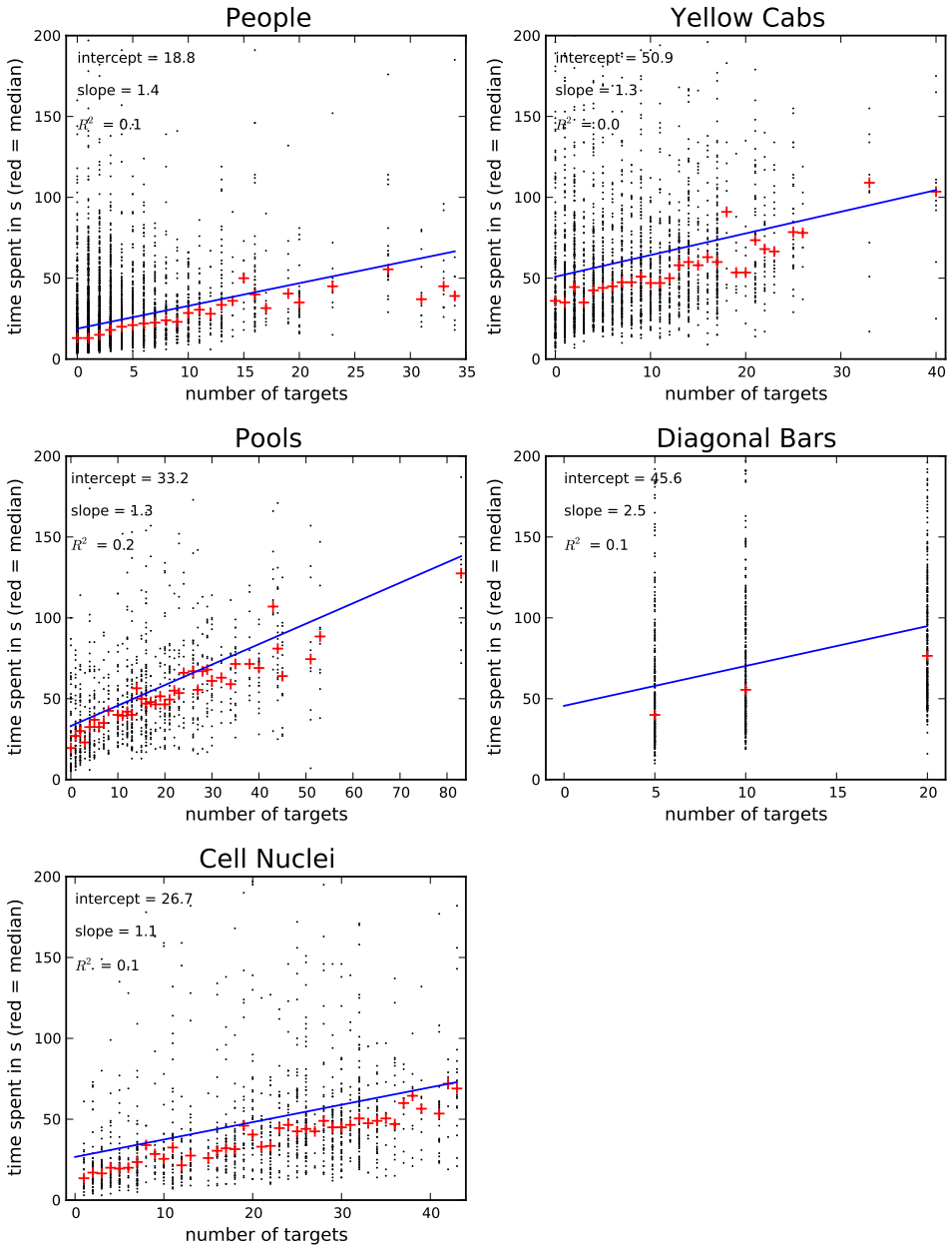

Figure 6.16: The time spent per image versus the number of targets in the image. Red marker is the median for a given number of targets. The parameters of the blue line fitted using least squares to the data is shown in the upper left of each plot.

10 or more images are shown in black.

6.8.8 Annotator Precision Recall

Figure 6.18 is the same as Figure 6.2, except that it shows annotator that labeled fewer than 10 images as red markers, and the others as black markers. This is to show that many of the worst annotators labeled very few images, and may have given up after that.

6.8.9 Time Spent in Iterative Task

The distribution of time spent per image for the iterative task. Figure 6.19 provides more information than the bottom diagram of Figure 6.9. The first row shows the first iteration after the "seed" task, the second row shows the second iteration, and so on until the fifth iteration.

6.8.10 Time Spent For Varying Target Difficulty

The diagonal bars dataset was generated with three levels of difficulty. The target bars could be oriented at either 10°, 15°, or 30° degrees off the diagonal. All bars in an image were of the same difficulty, but the number of bars were varied at 5, 10 and 20 bars per image. The histograms in Figure 6.20 show the distribution of the time spent for different levels of target difficulty, the top histogram showing the most difficult task. It is clear that our annotators were faster when the task was easier.

6.8.11 Cell Nuclei Counting: Experts versus Turkers

Pathologists and biologists are often interested in counting the number of cells in a particular sample image. The plots in Figure 6.21 look at how well the annotators perform on this task. The horizontal axis in the plots below show the number of cell detections made in each image for one of the expert pathologists. For the first three images, the annotators' detections were clustered, and a cluster was counted

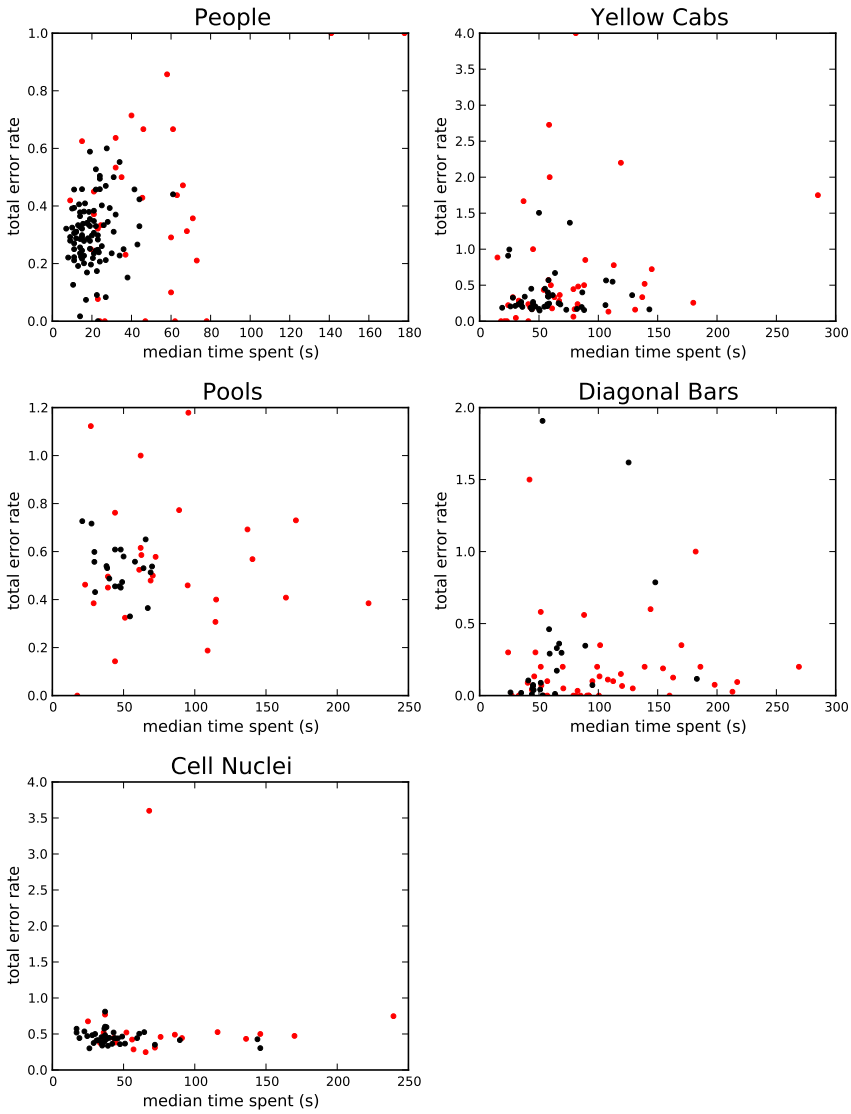

Figure 6.17: Each marker represents a different annotator, with the horizontal axis being the median time spent and the vertical axis being the total error rate. The annotators that annotated fewer than 10 images are shown as red dots while annotators that annotated 10 or more images are shown in black.

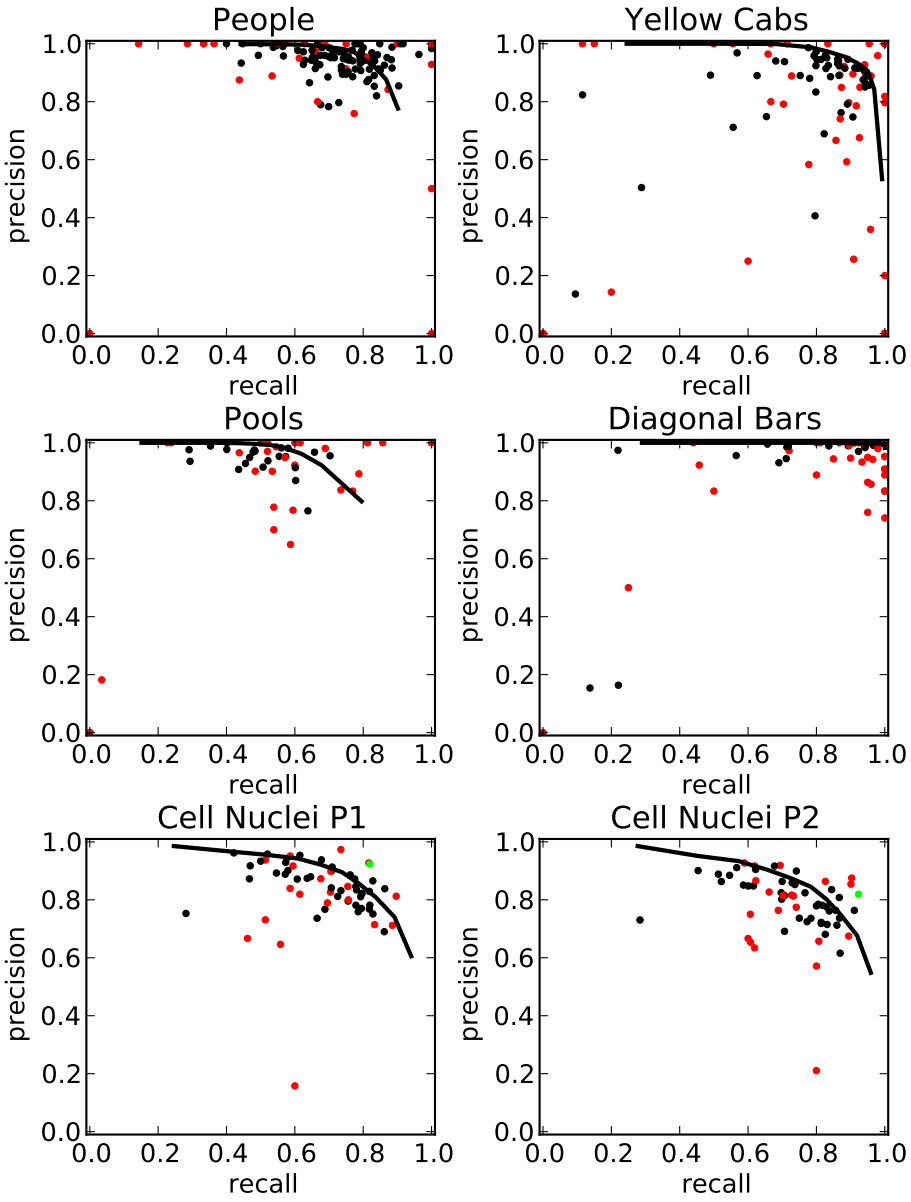

Figure 6.18: Precision and recall for each annotator. Same data as in Figure 6.2. Only difference is in the presentation: annotator that labeled fewer than 10 images as red markers, and the others as black markers.

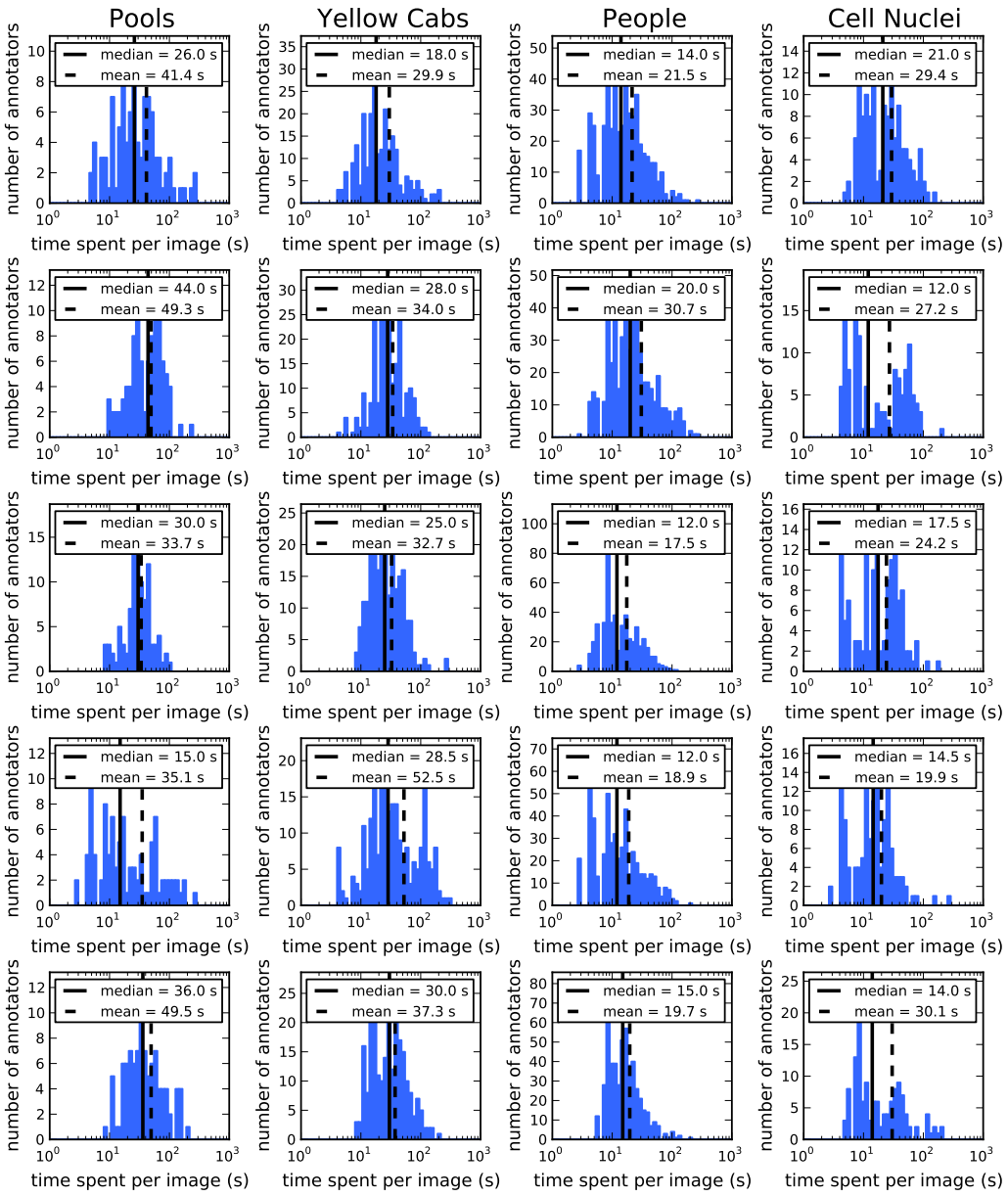

Figure 6.19: Time spent per image in the iterative annotation task. The first row show the first iteration after the "seed" task, the second row shows the second iteration, and so on until the fifth iteration.

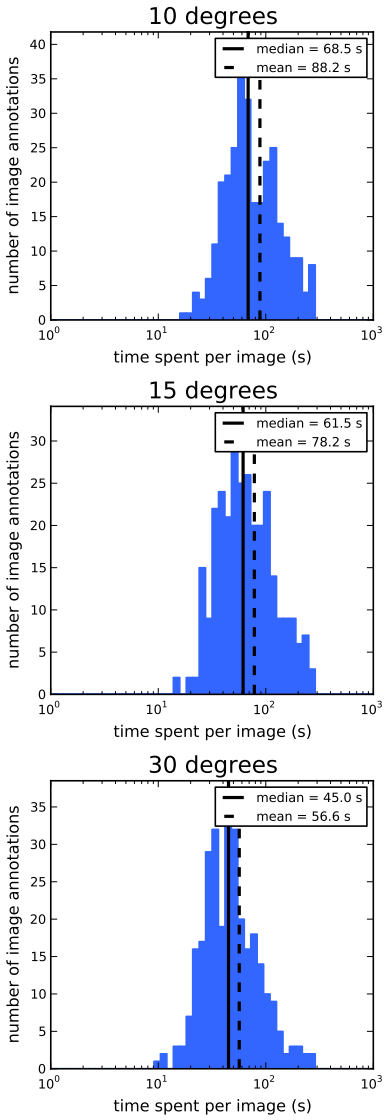

Figure 6.20: Time spent per image for different levels of target ambiguity.

as a detection if it had at least n number of votes (a cluster could at most have 10 votes). The number of votes needed, n, is varied in the three first plots. In the forth plot, the first expert is plotted against the number of detections per image provided by the second expert. As a sanity check, in the fifth plot the second expert is on the horizontal axis, and the annotator counts with two votes are on the vertical axis. Random jitter of up to 0.3 counts was added to separate overlapping markers.

6.9 Appendix: Detection Clustering

The matching algorithm is described in the main paper. This section describes a simple heuristic clustering algorithm for clustering the detections.

Let $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ be a list of detection lists provided by different annotators, where $X_p = \{x_{p1}, x_{p2}, \ldots, x_{ps}\}$ is a list of detections from the pth annotator. Each detection is a D-dimensional point, $x_{pi} \in \mathbb{R}^D$ (in our experiments $D = 2$ always). Let $C = \{c_1, \ldots, c_m\}$ be a list of cluster centers. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be a list of lists of cluster assignments. A_q is a list of the detections in cluster q.

1. Sort the lists in $\mathcal{X}$ by the size of each list, with the largest first. Denote the sorted list of lists of detections $\widehat{\mathcal{X}}$.

2. Initialize the cluster centers C and the cluster assignment lists $\mathcal{A}$ with the detections from the first list in $\widehat{\mathcal{X}}$ (the list with most detections).

3. For each of the *remaining* lists $X_p \in \widehat{\mathcal{X}}$, match the detections from list X_p to the cluster centers in C and assign them to the corresponding assignment lists in $\mathcal{A}$:

   (a) For each detection $x_{pi} \in X_p$, find the nearest neighbor cluster r (with center c_r) with no detections from the current detection list X_p assigned to it.

   (b) If the distance to the nearest neighbor is greater than a threshold, $d(x_{pi}, c_r) > \tau$, then assign the detection to a new cluster. Add the new cluster center

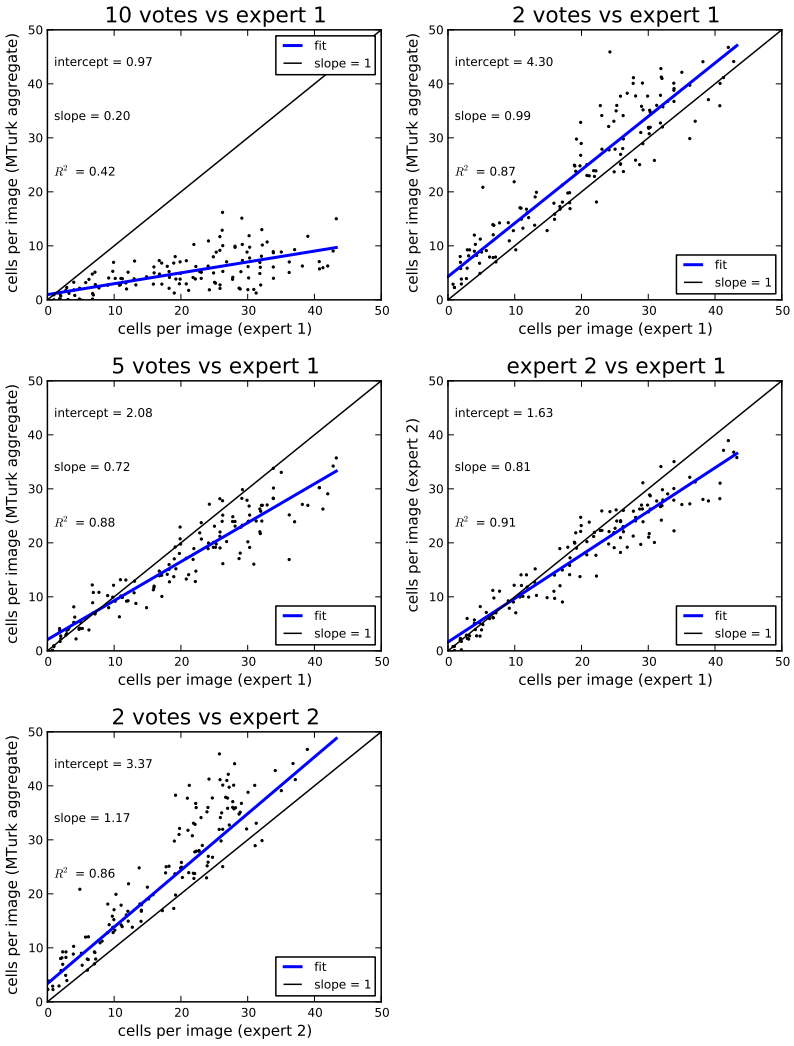

Figure 6.21: Comparing non-expert MTurk annotators to experts. The plots show the number of cells detected by a consensus of MTurk annotators with that of expert 1 or expert 2 on the cell nuclei dataset.

to C and add a new assignment list to $\mathcal{A}$ containing only x_{pi}. Process the next detection.

(c) Otherwise, check if the nearest neighbor cluster r has other detections in X_p that are closer than x_{pi}. If there are no such detections, then assign x_{pi} to cluster r, i.e. add x_{pi} to A_r.

(d) If a closer detection x_{pj} was found, i.e. $d(x_{pj}, c_r) < d(x_{pi}, c_r)$, then assign x_{pj} to cluster r. Go back to (a) and try to assign x_{pi} again.

(e) Every time a detection is assigned to a cluster, the center of the cluster is recomputed by taking the mean location of all assigned detections.

Chapter 7

Active Labeling

7.1 Abstract

Labeling large datasets has become faster, cheaper, and easier with the advent of crowdsourcing services like Amazon Mechanical Turk. How can one trust the labels obtained from such services? We propose a model of the labeling process which includes label uncertainty, as well a multi-dimensional measure of the annotators' ability. From the model we derive an online algorithm that estimates the most likely value of the labels and the annotator abilities. It finds and prioritizes experts when requesting labels, and actively excludes unreliable annotators. Based on labels already obtained, it dynamically chooses which images will be labeled next, and how many labels to request in order to achieve a desired level of confidence. Our algorithm is general and can handle binary, multi-valued, and continuous annotations (e.g. bounding boxes). Experiments on a dataset containing more than 50,000 labels show that our algorithm reduces the number of labels required, and thus the total cost of labeling, by a large factor while keeping error rates low on a variety of datasets.

7.2 Introduction

Crowdsourcing, the act of outsourcing work to a large crowd of workers, is rapidly changing the way datasets are created. Not long ago, labeling large datasets could take weeks, if not months. It was necessary to train annotators on custom-built

interfaces, often in person, and to ensure they were motivated enough to do high quality work. Today, with services such as Amazon Mechanical Turk (MTurk), it is possible to assign annotation jobs to hundreds, even thousands, of computer-literate workers and get results back in a matter of hours. This opens the door to labeling huge datasets with millions of images, which in turn provides great possibilities for training computer vision algorithms.

The quality of the labels obtained from annotators varies. Some annotators provide random or bad quality labels in the hope that they will go unnoticed and still be paid, and yet others may have good intentions but completely misunderstand the task at hand. The standard solution to the problem of "noisy" labels is to assign the same labeling task to many different annotators, in the hope that at least a few of them will provide high quality labels or that a consensus emerges from a great number of labels. In either case, a large number of labels is necessary, and although a single label is cheap, the costs can accumulate quickly.

If one is aiming for a given label quality for the minimum time and money, it makes more sense to dynamically decide on the number of labelers needed. If an expert annotator provides a label, we can probably rely on it being of high quality, and we may not need more labels for that particular task. On the other hand, if an unreliable annotator provides a label, we should probably ask for more labels until we find an expert or until we have enough labels from non-experts to let the majority decide the label.

We present an online algorithm to estimate the reliability or expertise of the annotators, and to decide how many labels to request per image based on who has labeled it. The model is general enough to handle many different types of annotations, and we show results on binary, multi-valued, and continuous-valued annotations collected from MTurk.

The general annotator model is presented in Section 7.4 and the online version in Section 7.5. Adaptations of the model to discrete and continuous annotations are discussed in Section 7.6. The datasets are described in Section 7.7, the experiments in Section 7.8, and we conclude in Section 7.9.

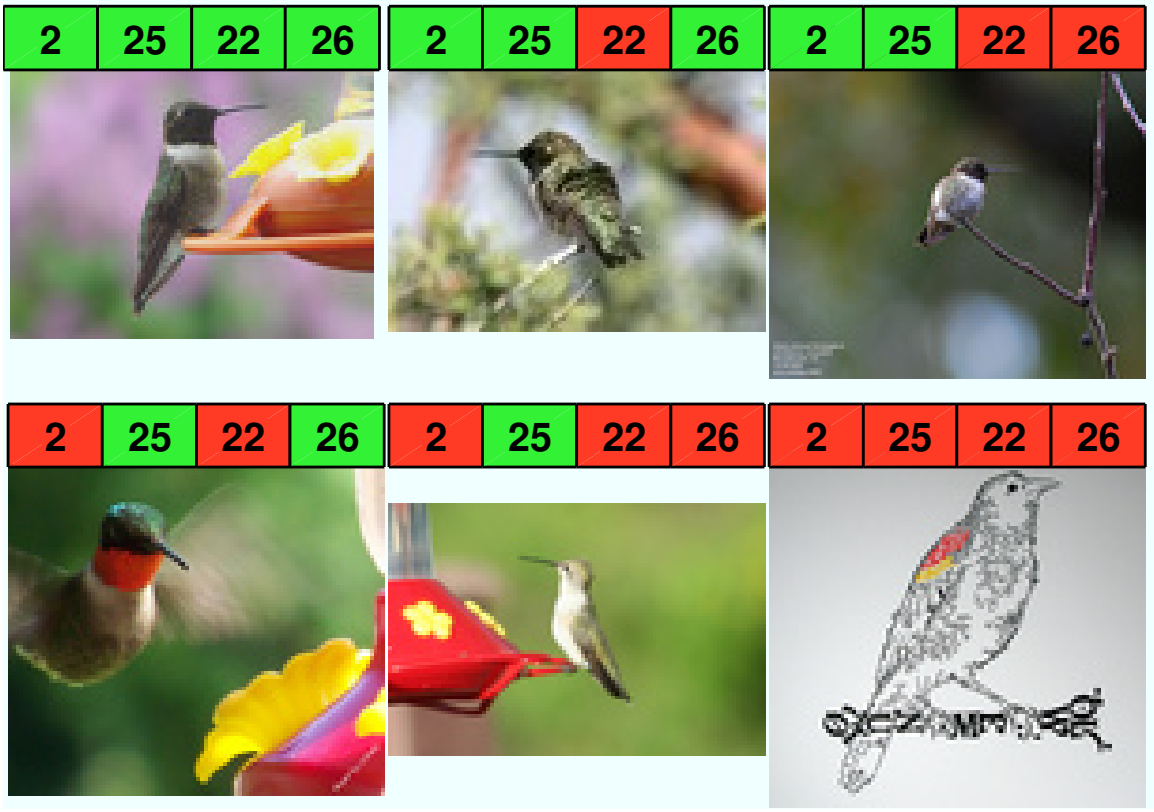

Figure 7.1: Examples of binary labels obtained from Amazon Mechanical Turk, (see Figure 7.2 for an example of continuous labels). The boxes show the labels provided by four workers (identified by the number in each box); green indicates that the worker selected the image, red means that he or she did not. The task for each annotator was to select only images that he or she thought contained a Black-chinned Hummingbird. Figure 7.5 shows the expertise and bias of the workers. Worker 25 has a high false positive rate, and 22 has a high false negative rate. Worker 26 provides inconsistent labels, and 2 is the annotator with the highest accuracy. Photos in the top row were classified to contain a Black-chinned Hummingbird by our algorithm, while the ones in the bottom row were not.

7.3 Related Work

The quality of crowdsourced labels (also called annotations or tags) has been studied before in different domains. In computer vision, the quality of annotations provided by MTurk workers and by volunteers has been explored for a wide range of annotation types [SF08, RTMF08]. In natural language processing, Snow et al. [SOJN08] gathered labels from MTurk and compared the quality to ground truth.

The most common method for obtaining ground truth annotations from crowdsourced labels is by applying a majority consensus heuristic. This has been taken one step further by looking at the consistency between labelers. For multi-valued annotations, Dawid and Skene [DS79] modeled the individual annotator accuracy by a confusion matrix. Sheng et al. [SPI08] also modeled annotator quality, and showed how repeated and selective labeling increased the overall labeling quality on synthetic data. Smyth et al. [SFB+95] integrated the opinions of many experts to determine a gold standard, and Spain and Perona [SP08] developed a method for combining prioritized lists obtained from different annotators. Using annotator consistency to obtain ground truth has also been used in the context of paired games and CAPTCHAs [vAD04, vAMM+08]. Whitehill et al. [WRW+09] consider the difficulty of the task and the ability of the annotators. Annotator models have also been used to train classifiers with noisy labels [RYZ+09].

Vijayanarasimhan and Grauman [VG09] proposed a system which actively asks for image labels that are the most informative and cost effective. To our knowledge, the problem of online estimation of annotator reliabilities has not been studied before.

7.4 Modeling Annotators and Labels

We assume that each image i has an unknown "target value" which we denote by $\mathbf{z}_i$. This may be a continuous or discrete scalar or vector. The set of all N images, indexed by image number, is $\mathcal{I} = \{1, \ldots, N\}$, and the set of corresponding target values is abbreviated $\mathbf{z} = \{\mathbf{z}_i\}_{i=1}^{N}$. The reliability or expertise of annotator j is

described by a vector of parameters, $\mathbf{a}_j$. For example, it can be scalar, $\mathbf{a}_j = a_j$, such as the probability that the annotator provides a correct label; specific annotator parameterizations are discussed in Section 7.6. There are M annotators in total, $\mathcal{A} = \{1, \ldots, M\}$, and the set of their parameter vectors is $\mathbf{a} = \{\mathbf{a}_j\}_{j=1}^{M}$. Each annotator j provides labels $\mathcal{L}^j = \{\mathbf{l}_{ij}\}_{i \in \mathcal{I}_j}$ for all or a subset of the images, $\mathcal{I}_j \subseteq \mathcal{I}$. Likewise, each image i has labels $\mathcal{L}_i = \{\mathbf{l}_{ij}\}_{j \in \mathcal{A}_i}$ provided by a subset of the annotators $\mathcal{A}_i \subseteq \mathcal{A}$. The set of all labels is denoted $\mathcal{L}$. For simplicity, we will assume that the labels $\mathbf{l}_{ij}$ belong to the same set as the underlying target values $\mathbf{z}_i$; this assumption could, in principle, be relaxed.

Our causal model of the labeling process is shown schematically in Figure 7.4. The image target values and annotator parameters are assumed to be generated independently. To ensure that the estimation process degrades gracefully with little available label data, we take the Bayesian point of view with priors on $\mathbf{z}_i$ and $\mathbf{a}_j$ parameterized by $\boldsymbol{\zeta}$ and $\boldsymbol{\alpha}$, respectively. The priors encode our prior belief of how skilled the annotators are (e.g. that half will be experts and the rest unskilled), and what kind of target values we expect. The joint probability distribution can thus be factorized as

$$p(\mathcal{L}, \mathbf{z}, \mathbf{a}) = \prod_{i=1}^{N} p(\mathbf{z}_i \mid \boldsymbol{\zeta}) \prod_{j=1}^{M} p(\mathbf{a}_j \mid \boldsymbol{\alpha}) \prod_{\mathbf{l}_{ij} \in \mathcal{L}} p(\mathbf{l}_{ij} \mid \mathbf{z}_i, \mathbf{a}_j). \tag{7.1}$$

Inference: Given the observed variables, that is, the labels $\mathcal{L}$, we would like to infer the hidden variables, i.e. the target values $\mathbf{z}$, as well as the annotator parameters $\mathbf{a}$. This can be done using a Bayesian treatment of the Expectation-Maximization (EM) algorithm [DLR$^+$77].

E-step: Assuming that we have a current estimate $\hat{\mathbf{a}}$ of the annotator parameters, we compute the posterior on the target values:

$$\hat{p}(\mathbf{z}) = p(\mathbf{z} \mid \mathcal{L}, \hat{\mathbf{a}}) \propto p(\mathbf{z})\, p(\mathcal{L} \mid \mathbf{z}, \hat{\mathbf{a}}) = \prod_{i=1}^{N} \hat{p}(\mathbf{z}_i), \tag{7.2}$$

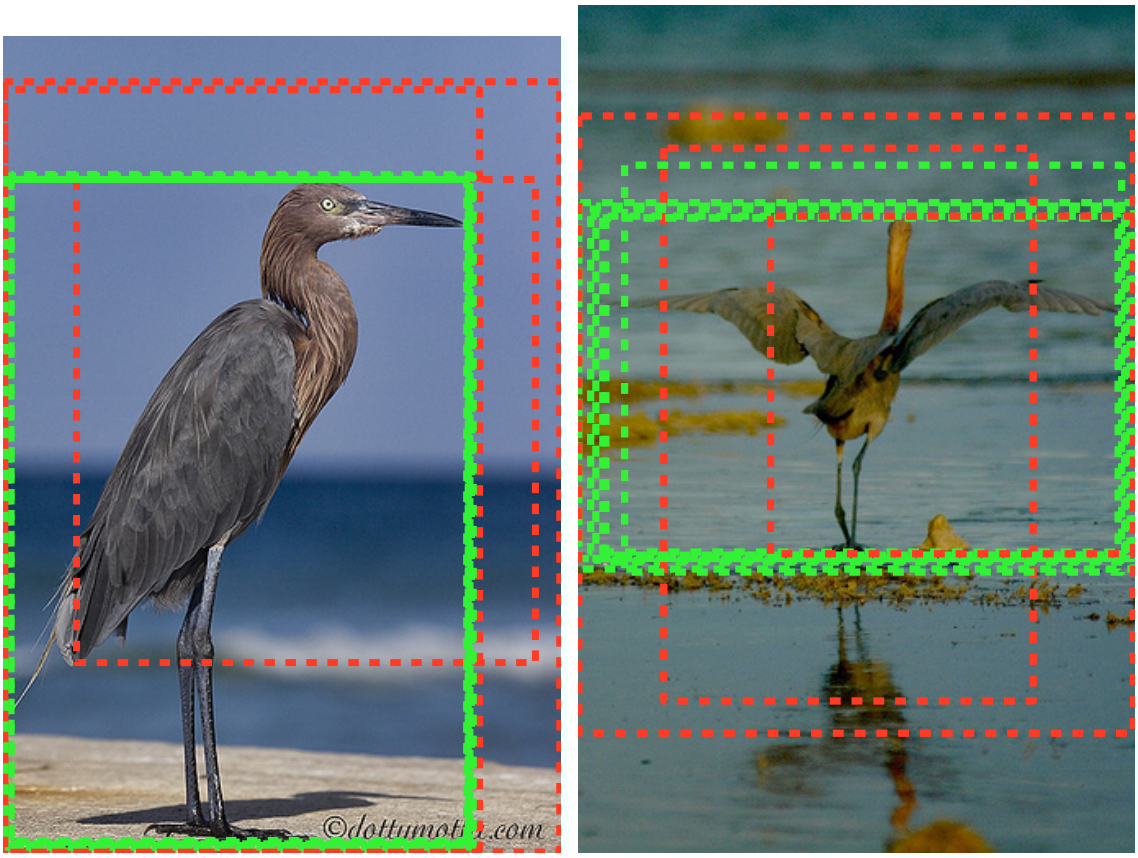

Figure 7.2: Examples of bounding boxes (10 per image) obtained from MTurk workers who were instructed to provide a snug fit. Per our model, the green boxes are correct and the red boxes incorrect. Most workers provide consistent labels, but two unreliable workers stand out: no. 53 and 58 (they provided two of the incorrect boxes in each image). As can be seen from Figure 7.6c, most of the labels provided by these two workers were of low quality.

where

$$\hat{p}(\mathbf{z}_i) = p(\mathbf{z}_i \mid \boldsymbol{\zeta}) \prod_{j \in \mathcal{A}_i} p(\mathbf{l}_{ij} \mid \mathbf{z}_i, \hat{\mathbf{a}}_j). \tag{7.3}$$

M-step: To estimate the annotator parameters $\mathbf{a}$, we maximize the expectation of the logarithm of the posterior on $\mathbf{a}$ with respect to $\hat{p}(\mathbf{z}_i)$ from the E-step. We call the auxiliary function being maximized $Q(\mathbf{a}, \hat{\mathbf{a}})$. Thus the optimal $\mathbf{a}^\star$ is found from

$$\mathbf{a}^\star = \arg\max_{\mathbf{a}} Q(\mathbf{a}, \hat{\mathbf{a}}), \tag{7.4}$$

where $\hat{\mathbf{a}}$ is the estimate from the previous iteration, and

$$Q(\mathbf{a}, \hat{\mathbf{a}}) = \mathbb{E}_{\mathbf{z}} \left[\log p(\mathcal{L} \mid \mathbf{z}, \mathbf{a}) + \log p(\mathbf{a} \mid \boldsymbol{\alpha}) \right] \tag{7.5}$$

$$= \sum_{j=1}^{M} Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j), \tag{7.6}$$

where $\mathbb{E}_{\mathbf{z}}[\cdot]$ is the expectation with respect to $\hat{p}(\mathbf{z})$ and $Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j)$ is defined as

$$Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j) = \log p(\mathbf{a}_j \mid \boldsymbol{\alpha}) + \sum_{i \in \mathcal{I}_j} \mathbb{E}_{\mathbf{z}_i} \left[\log p(\mathbf{l}_{ij} \mid \mathbf{z}_i, \mathbf{a}_j) \right]. \tag{7.7}$$

Hence, the optimization can be carried out separately for each annotator, and relies only on the labels that the annotator provided. It is clear from the form of (7.3) and (7.7) that any given annotator is not required to label every image.

7.5 Online Estimation

The factorized form of the general model in (7.1) allows for an online implementation of the EM-algorithm. Instead of asking for a fixed number of labels per image, the online algorithm actively asks for labels only for images where the target value is still uncertain. Furthermore, it finds and prioritizes expert annotators and blocks sloppy annotators online. The algorithm is outlined in Figure 7.5 and discussed in detail in the following paragraphs.

Initially, we are faced with a set of images $\mathcal{U}$ with unknown target values $\mathbf{z}$. The

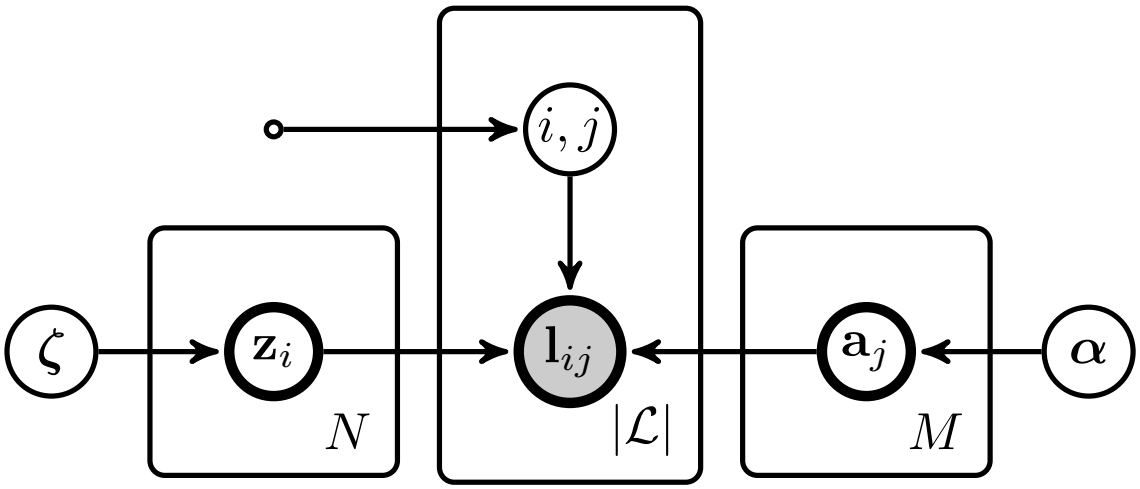

Figure 7.3: Plate representation of the general model. The i, j pair in the middle plate, indicating which images each annotator labels, is determined by some process that depends on the algorithm (see Sections 7.4–7.5).

Input: Set of images $\mathcal{U}$ to be labeled

1: Initialize $\mathcal{I}, \mathcal{L}, \mathcal{E}, \mathcal{B} = \{\emptyset\}$
2: **while** $|\mathcal{I}| < |\mathcal{U}|$ **do**
3: Add n images $\{i : i \in (\mathcal{U} \setminus \mathcal{I})\}$ to $\mathcal{I}$
4: **for** $i \in \mathcal{I}$ **do**
5: Compute $\hat{p}(\mathbf{z}_i)$ from $\mathcal{L}_i$ and $\mathbf{a}$
6: **while** $\max_{\mathbf{z}_i} \hat{p}(\mathbf{z}_i) < \tau$ and $|\mathcal{L}_i| < m$ **do**
7: Ask expert annotators $j \in \mathcal{E}$ to provide a label $\mathbf{l}_{ij}$
8: **if** no label $\mathbf{l}_{ij}$ is provided within time T **then**
9: Obtain label $\mathbf{l}_{ij}$ from some annotator $j \in (\mathcal{A}' \setminus \mathcal{B})$
10: $\mathcal{L}_i \leftarrow \{\mathcal{L}_i \cup \mathbf{l}_{ij}\}$, $\mathcal{A} \leftarrow \{\mathcal{A} \cup \{j\}\}$
11: Recompute $\hat{p}(\mathbf{z}_i)$ from updated $\mathcal{L}_i$ and $\mathbf{a}$
12: Set $\mathcal{E}, \mathcal{B} = \{\emptyset\}$
13: **for** $j \in \mathcal{A}$ **do**
14: Estimate $\mathbf{a}_j$ from $\hat{p}(\mathbf{z}_i)$ by $\max_{\mathbf{a}_j} Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j)$
15: **if** $\text{var}(\mathbf{a}_j) < \theta_v$ **then**
16: **if** $\mathbf{a}_j$ satisfies an expert criterion **then**
17: $\mathcal{E} \leftarrow \{\mathcal{E} \cup \{j\}\}$
18: **else**
19: $\mathcal{B} \leftarrow \{\mathcal{B} \cup \{j\}\}$
20: Output $\hat{p}(\mathbf{z})$ and $\mathbf{a}$

Figure 7.4: Online algorithm for estimating annotator parameters and actively choosing which images to label. The label collection step is outlined on lines 3–11, and the annotator evaluation step on lines 12–19. See Section 7.5 for details.

set $\mathcal{I} \subseteq \mathcal{U}$ denotes the set of images for which at least one label has been collected and $\mathcal{L}$ is the set of all labels provided so far. Initially $\mathcal{I}$ and the set of all labels $\mathcal{L}$ are empty. We assume that there is a large pool of annotators $\mathcal{A}'$, of different and unknown ability, available to provide labels. The set of annotators that have provided labels so far is denoted $\mathcal{A} \subseteq \mathcal{A}'$ and is initially empty. We keep two lists of annotators: the *expert*-list, $\mathcal{E} \subseteq \mathcal{A}$, is a set of annotators who we trust to provide good labels, and the *bot*-list, $\mathcal{B} \subseteq \mathcal{A}$, are annotators that we know provide low quality labels and would like to exclude from further labeling. We call the latter list "bot"-list because the labels could as well have been provided by a robot choosing labels at random. The algorithm proceeds by iterating two steps until all the images have been labeled: (1) the label collection step, and (2) the annotator evaluation step.

Label collection step: $\mathcal{I}$ is expanded with n new images from $\mathcal{U}$. Next, the algorithm asks annotators to label the images in $\mathcal{I}$. First annotators in $\mathcal{E}$ are asked. If no annotator from $\mathcal{E}$ is willing to provide a label within a fixed amount of time T, the label is instead collected from an annotator in $(\mathcal{A}' \setminus \mathcal{B})$. For each image $i \in \mathcal{I}$, new labels $\mathbf{l}_{ij}$ are requested until either the posterior on the target value $\mathbf{z}_i$ is above a confidence threshold τ,

$$\max_{\mathbf{z}_i} \hat{p}(\mathbf{z}_i) > \tau, \tag{7.8}$$

or the number of labels $|\mathcal{L}_i|$ has reached a maximum cutoff m. It is also possible to set different thresholds for different $\mathbf{z}_i$'s, in which case we can trade off the costs of different kinds of target value misclassifications. The algorithm proceeds to the annotator evaluation step.

Annotator evaluation step: Since posteriors on the image target values $\hat{p}(\mathbf{z}_i)$ are computed in the label collection step, the annotator parameters can be estimated in the same manner as in the M-step in the EM-algorithm, by maximizing $Q_j(\mathbf{a}_j, \hat{\mathbf{a}}_j)$ in (7.7). Annotator j is put in either $\mathcal{E}$ or $\mathcal{B}$ if a measure of the variance of $\mathbf{a}_j$ is below a threshold,

$$\text{var}(\mathbf{a}_j) < \theta_v, \tag{7.9}$$

where θ_v is the threshold on the variance. If the variance is above the threshold

we do not have enough evidence to consider the annotator to be an expert or a bot (unreliable annotator). If the variance is below the threshold, we place the annotator in $\mathcal{E}$ if $\mathbf{a}_j$ satisfies some expert criterion based on the annotation type, otherwise the annotator will be placed in $\mathcal{B}$ and excluded labeling in the next iteration.

On MTurk the expert- and bot-lists can be implemented by using "qualifications". A qualification is simply a pair of two numbers, a unique qualification id number and a scalar qualification score, that can be applied to any worker. The qualifications can then be used to restrict (by inclusion or exclusion) which workers are allowed to work on a particular task.

7.6 Annotation Types

Binary annotations are often used for classification, such as "Does the image contain an object from the visual class X or not?". In this case, both the target value $\mathbf{z}_i$ and the label $\mathbf{l}_{ij}$ are binary (dichotomous) scalars that can take values $z_i, l_{ij} \in \{0, 1\}$. A natural parameterization of the annotators is in terms of true negative and true positive rates. That is, let $\mathbf{a}_j = (a_j^0, a_j^1)^{\mathrm{T}}$ be the vector of annotator parameters, where

$$\begin{aligned} p(l_{ij} = 1 \mid z_i = 1, \mathbf{a}_j) &= a_j^1, \\ p(l_{ij} = 0 \mid z_i = 0, \mathbf{a}_j) &= a_j^0. \end{aligned} \tag{7.10}$$

As a prior for $\mathbf{a}_j$ we chose a mixture of beta distributions,

$$p(a_j^0, a_j^1) = \sum_{k=1}^{K} \pi_k^a \operatorname{Beta}(\alpha_{k,0}^0, \alpha_{k,0}^1) \operatorname{Beta}(\alpha_{k,1}^0, \alpha_{k,1}^1). \tag{7.11}$$

Our prior belief of the number of different types of annotators is encoded in the number of components K. For example, we can assume $K = 2$ kinds of annotators: honest annotators of different grades (unreliable to experts) are modeled by Beta densities that are increasingly peaked towards one, and adversarial annotators who

provide labels that are opposite of the target value are modeled by Beta distributions that are peaked towards zero (we have actually observed such annotators). The prior also acts as a regularizer in the EM-algorithm to ensure we do not classify an annotator as an expert until we have enough evidence.

The parameterization of true positive and true negative rates allows us to cast the model in a signal detection theoretic framework [Wic02], which provides a more natural separation of annotator bias and accuracy. Assume a signal x_i is generated in the head of the annotator as a result of some neural processing when the annotator is looking at image i. If the signal x_i is above a threshold t_j, the annotator chooses $l_{ij} = 1$, otherwise picking $l_{ij} = 0$. If we assume that the signal x_i is a random variable generated from one of two distributions, $p(x_i \mid z_i = k) \sim \mathcal{N}(\mu_j^k, \sigma^2)$, we can compute the annotator's sensitivity index d'_j, defined as [Wic02],

$$d'_j = \frac{|\mu_j^1 - \mu_j^0|}{\sigma}. \tag{7.12}$$

Notice that d'_j is a quantity representing the annotator's ability to discriminate images belonging to the two classes, while t_j is a quantity representing the annotator's propensity towards label 1 (low t_j) or label 0 (high t_j). By varying t_j and recording the false positive and false negative rates, we get the receiver operating characteristic (ROC curve) of the annotator. When $t_j = 0$ then the annotator is unbiased and will produce equal false positive and negative error rates of 50%, 31%, 15%, and 6% for $d'_j = \{0, 1, 2, 3\}$ respectively. It is possible go between the two parameterizations if we assume that σ is the same for all annotators. For example, by assuming $\sigma = 1$, $\mu_j^0 = -d'_j/2$ and $\mu_j^1 = d'_j/2$, we can convert between the two representations using,

$$\begin{bmatrix} 1 & \frac{1}{2} \\ 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} t_j \\ d'_j \end{bmatrix} = \begin{bmatrix} \Phi^{-1}(a_j^0) \\ \Phi^{-1}(1 - a_j^1) \end{bmatrix}, \tag{7.13}$$

where $\Phi^{-1}(\cdot)$ is the inverse of the standard normal cumulative probability density function.

For binary labels, the stopping criterion in (7.8) has a very simple form. Consider

the logarithm of the ratio of the posteriors,

$$R_i = \log \frac{p(z_i = 1 \mid \mathcal{L}_i, \mathbf{a})}{p(z_i = 0 \mid \mathcal{L}_i, \mathbf{a})} = \log \frac{p(z_i = 1)}{p(z_i = 0)} + \sum_{l_{ij} \in \mathcal{L}_i} R_{ij}, \tag{7.14}$$

where $R_{ij} = \log \frac{p(l_{ij}|z_i=1,\mathbf{a}_j)}{p(l_{ij}|z_i=0,\mathbf{a}_j)}$. Thus, every label l_{ij} provided for image i by some annotator j adds another positive or negative term R_{ij} to the sum in (7.14). The magnitude $|R_{ij}|$ increases with d'_j, so that the opinions of expert annotators are valued more than unreliable ones. The criterion in (7.8) is equivalent to a criterion on the magnitude on the log ratio,

$$|R_i| > \tau' \text{ where } \tau' = \log \frac{\tau}{1-\tau}. \tag{7.15}$$

Observe that τ' could be different for positive and negative R_i. One would wish to have different thresholds if one had different costs for false alarm and false reject errors. In this case, the stopping criterion is equivalent to Wald's stopping rule for accepting or rejecting the null hypothesis in the Sequential Probability Ratio Test (SPRT) [Wal45].

To decide when we are confident in an estimate of $\mathbf{a}_j$, in the online algorithm, we estimate the variance $\text{var}(\mathbf{a}_j)$ by fitting a multivariate Gaussian to the peak of $p(\mathbf{a}_j \mid \mathcal{L}, \mathbf{z})$. As a criterion for expertise, i.e. whether to add annotator j to $\mathcal{E}$, we use $d'_j > 2$ corresponding to a 15% error rate.

Multi-valued annotations where $z_i, l_{ij} \in \{1, \ldots, D\}$, can be modeled in almost the same way as binary annotations. A general method is presented in [DS79] for a full confusion matrix. However, we used a simpler model where a single a_j parameterizes the ability of the annotator,

$$\begin{aligned} p(l_{ij} = z_i \mid a_j) &= a_j, \\ p(l_{ij} \neq z_i \mid a_j) &= \frac{1 - a_j}{D - 1}. \end{aligned} \tag{7.16}$$

Thus, the annotator is assumed to provide the correct value with probability a_j and an incorrect value with probability $(1-a_j)$. Using this parameterization, the methods described above can be applied to the multi-valued (polychtomous) case.

Continuous-valued annotations are also possible. To make this section concrete, and for simplicity of notation, we will use bounding boxes, see Figure 7.2. However, the techniques used here can be extended to other types of annotations, such as object locations, segmentations, ratings, etc.

The image labels and target values are the locations of the upper left (x_1, y_1) and lower right (x_2, y_2) corners of the bounding box, and thus $\mathbf{z}_i$ and $\mathbf{l}_{ij}$ are 4-dimensional vectors of continuous variables $(x_1, y_1, x_2, y_2)^{\mathrm{T}}$. The annotator behavior is assumed to be governed by a single parameter $a_j \in [0, 1]$, which is the probability that the annotator attempts to provide an honest label. The annotator provides a "random" bounding box with probability $(1 - a_j)$. An honest label is assumed to be normally distributed from the target value

$$p(\mathbf{l}_{ij} \mid \mathbf{z}_i) = \mathcal{N}(\mathbf{l}_{ij} \mid \mathbf{z}_i, \mathbf{\Sigma}), \tag{7.17}$$

where $\mathbf{\Sigma} = \sigma^2\mathbf{I}$ is assumed to be a diagonal. One can take the Bayesian approach and have a prior on σ, and let it vary for different images. However, for simplicity we choose to keep σ fixed at 4 pixels (in screen coordinates). If the annotator decides not to provide an honest label, the label is assumed to be drawn from a uniform distribution,

$$p(\mathbf{l}_{ij} \mid \mathbf{z}_i) = \lambda_i^{-2}, \tag{7.18}$$

where λ_i is the area of image i (other variants, such as a very broad Gaussian, are also possible). The posterior on the label, used in the E-step in (7.2), can thus be written as a mixture,

$$p(\mathbf{l}_{ij} \mid \mathbf{z}_i, a_j) = a_j\mathcal{N}(\mathbf{l}_{ij} \mid \mathbf{z}_i, \mathbf{\Sigma}) + (1 - a_j)\frac{1}{\lambda^2}. \tag{7.19}$$

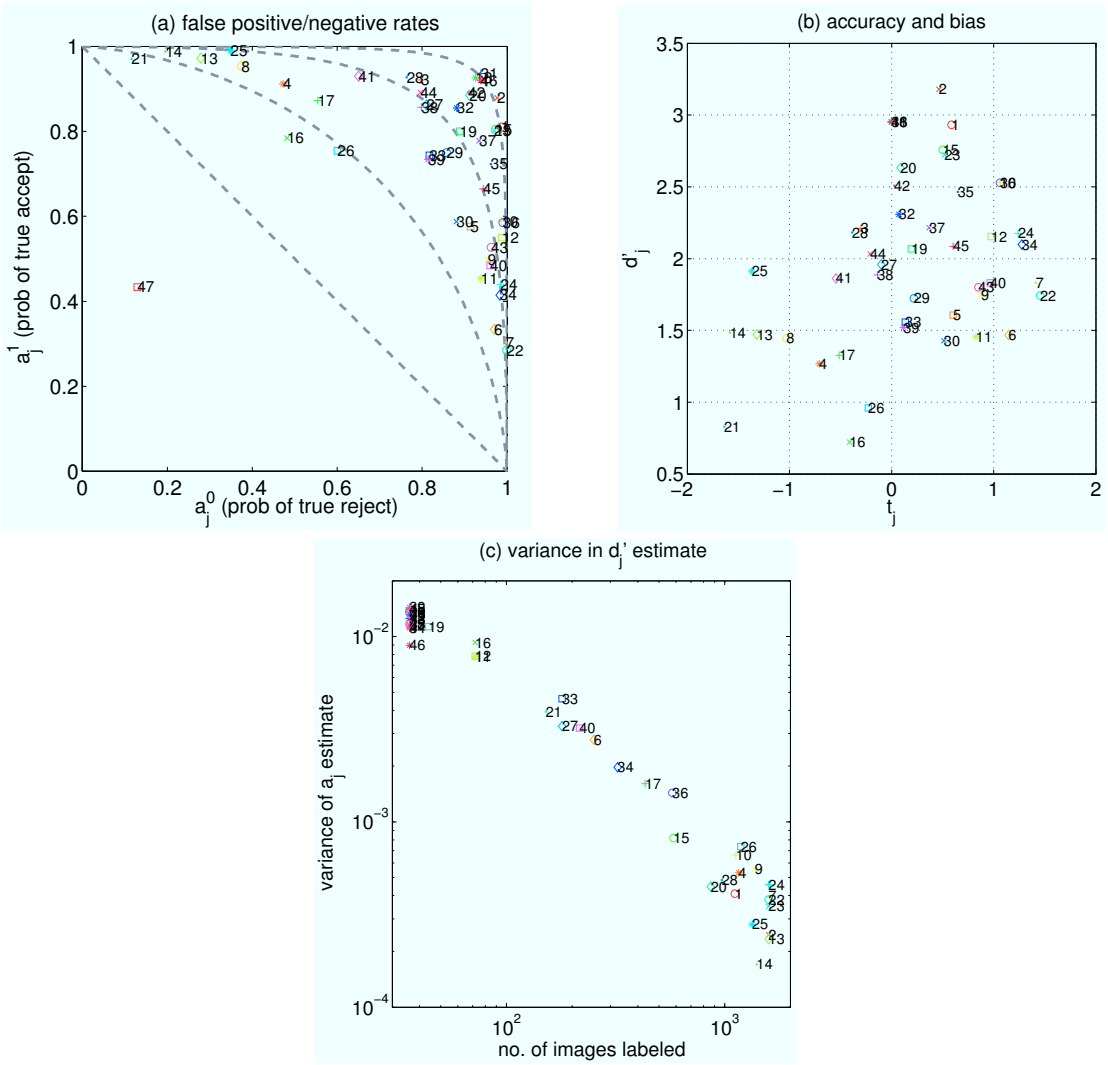

Figure 7.5: Estimates of expertise and bias in annotators providing binary labels. Annotators are plotted with different symbols and numbers to make them easier to locate across the plots. (a) Estimated true negative and positive rates (a_j^0, a_j^1). The dotted curves show the ROC curves for $d'_j = \{0, 1, 2, 3\}$. (b) Estimated t_j and d'_j from the data in (a). The accuracy of the annotator increases with d'_j and the bias is reflected in t_j. For example, if t_j is positive, the annotator has a high correct rejection rate at the cost of some false rejections, see Figure 7.1 for some specific examples. The outlier annotator, no. 47 in (a), with negative d'_j, indicating adversarial labels, was excluded from the plot. (c) The variance of the (a_j^0, a_j^1) decreases quickly with the number of images labeled. These diagrams show the estimates for the Presence-1 workers; Presence-2 gave very similar results.

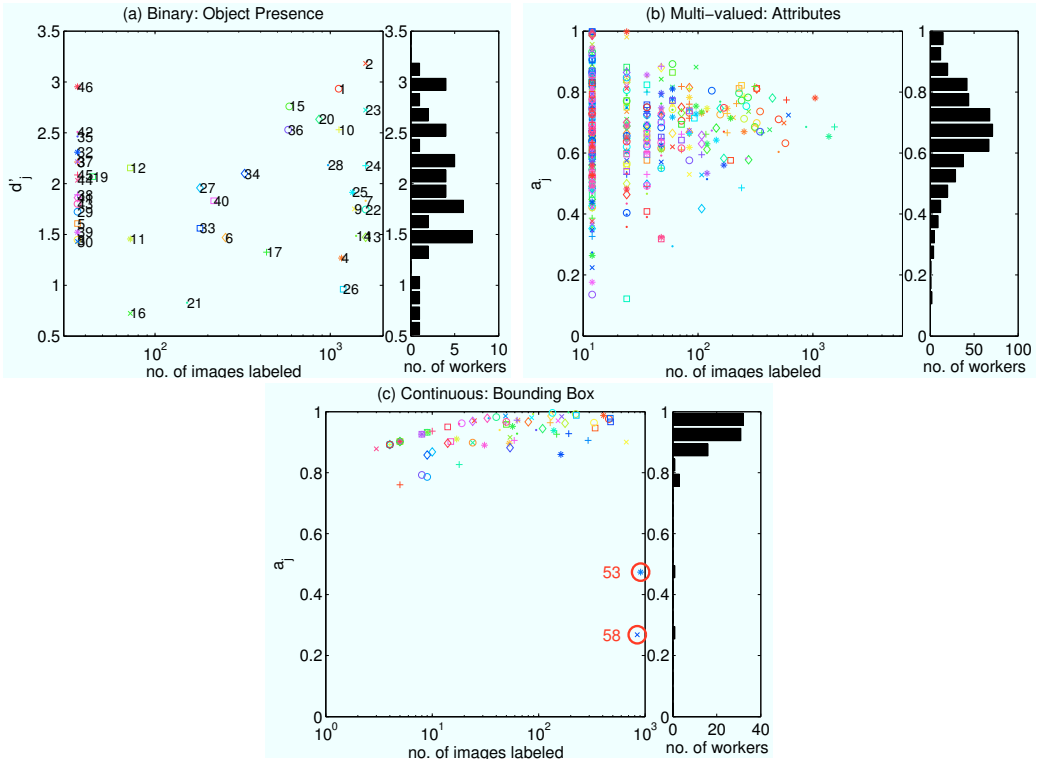

Figure 7.6: Quality index for each annotator vs. number of labeled images across the three annotation types. Annotators are plotted with different symbols and colors for easier visual separation. The bar chart to the right of each scatter plot is a histogram of the number of workers with a particular accuracy. (a) Object Presence: Shows results for Presence-1, Presence-2 is very similar. The minimum number of images a worker can label is 36, which explains the group of workers near the left edge. The adversarial annotator, no. 47, provided 36 labels and is not shown. (b) Attributes: results on Attributes-1. The corresponding plot for Attributes-2 is very similar. (c) Bounding box: Note that only two annotators, 53 and 58, labeled all 911 images. They also provided consistently worse labels than the other annotators. Figure 7.2 shows examples of the bounding boxes they provided.

The prior on $\mathbf{z}_i$ is modeled by a uniform distribution over the image area, $p(\mathbf{z}_i) = \lambda_i^{-2}$, implying that we expect bounding boxes anywhere in the image. Similarly to the binary case, the prior on a_j is modeled as a Beta mixture,

$$p(a_j) = \sum_{k=1}^{K} \pi_k^a \text{Beta}(\alpha_k^0, \alpha_k^1), \tag{7.20}$$

to account for at different groups of annotators of different skills. We used two components, one for experts (peaked at high a_j) and another for unreliable annotators (broader, and peaked at a lower a_j).

In the EM-algorithm we approximate the posterior on $\mathbf{z}_i$ by a delta function,

$$\hat{p}(\mathbf{z}_i) = p(\mathbf{z}_i \mid \mathcal{L}_i, \hat{a}_j) = \delta(\hat{\mathbf{z}}_i), \tag{7.21}$$

where $\hat{\mathbf{z}}_i$ is the best estimate of $\mathbf{z}_i$, to avoid slow sampling to compute the expectation in the E-step. This approach works well in practice since $\hat{p}(\mathbf{z}_i)$ is usually very peaked around a single value of $\mathbf{z}_i$.

7.7 Datasets

Object Presence: To test the general model applied to binary annotations, we asked workers on MTurk to select images if they thought the image contained a bird of a certain species, see Figure 7.1. The workers were shown a few example illustrations of birds of the species in different poses. We collected labels for two different bird species, Presence-1 (Black-chinned Hummingbird) and Presence-2 (Reddish Egret), summarized in Table 7.1.

Attributes: As an example of a multi-valued annotation, we asked workers to pick one out of D mutually exclusive choices for the shape of a bird shown in a photograph (Attributes-1, $D = 14$) and for the color pattern of its tail (Attributes-2, $D = 4$). We obtained 5 labels per image for a total of 6,033 images, see Table 7.1.

Bounding Boxes: The workers were asked to draw a tightly fitting bounding

box around the bird in each image (details in Table 7.1). Although it is possible to extend the model to multiple boxes per image, we ensured that there was exactly one bird in each image to keep things simple. See Figure 7.2 for some examples.

7.8 Experiments and Discussion

To establish the skills of annotators on MTurk, we applied the general annotator model to the datasets described in Section 7.7 and Table 7.1. We first estimated $\mathbf{a}_j$ on the full datasets (which we call *batch*). We then estimated both the $\mathbf{a}_j$ and $\mathbf{z}_i$ using the online algorithm, as described in the last part of this section.

Annotator bias: The results of the batch algorithm applied to the Presence-1 dataset is shown in Figure 7.5. Different annotators fall on different ROC curves, with a bias towards either more false positives or false negatives. This is even more explicit in Figure 7.5b, where d'_j is a measure of expertise and t_j of the bias. What is clear from these figures is that most annotators, no matter their expertise, have some bias. Examples of bias for a few representative annotators and images are shown in Figure 7.1. Bias is something to keep in mind when designing annotation tasks, as the wording of a question presumably influences workers. In our experiments most the annotators seemed to prefer false negatives to false positives.

Annotator accuracy: Figure 7.6 shows how the accuracy of MTurk annotators varies with the number of images they label for different annotation types. For the Presence-1 dataset, the few annotators that labeled most of the available images had very different d'_j. For Attributes-1, on the other hand, the annotators that labeled most images have very similar a_j. In the case of the bounding box annotations, most annotators provided good labels, except for no. 53 and 58. These two annotators were also the only ones to label all available images. In all three subplots of Figure 7.6, most workers provide only a few labels, and only some very active annotators label more than 100 images. Our findings in this figure are very similar to the results presented in Figure 6 of [SOJN08].

Importance of discrimination: The results in Figure 7.6 point out the impor-

Dataset	Images	Assignments	Workers
Presence-1	1,514	15	47
Presence-2	2,401	15	54
Attributes-1	6,033	5	507
Attributes-2	6,033	5	460
Bounding Boxes	911	10	85

Table 7.1: Summary of the datasets collected from Amazon Mechanical Turk showing the number of images per dataset, the number of labels per image (assignments), and total number of workers that provided labels. Presence-1/2 are binary labels, and Attributes-1/2 are multi-valued labels.

tance of online estimation of $\mathbf{a}_j$ and the use of expert- and bot-lists for obtaining labels on MTurk. The expert-list is needed to reduce the number of labels per image, as we can be more sure of the quality of the labels received from experts. Furthermore, without the expert-list to prioritize which annotators to ask first, the image will likely be labeled by a new worker, and thus the estimate of $\mathbf{a}_j$ for that worker will be very uncertain. The bot-list is needed to discriminate against sloppy annotators that will otherwise annotate most of the dataset in hope to make easy money, as shown by the outliers (no. 53 and 58) in Figure 7.6c.

Performance of binary model: We compared the performance of the annotator model applied to binary data, described in Section 7.6, to two other models of binary data, as the number of available labels per image, m, varied. The first method was a simple majority decision rule and the second method was the GLAD-algorithm presented in [WRW$^+$09]. Since we did not have access to the ground truth labels of the datasets, we generated synthetic data, where we knew the ground truth, as follows: (1) We used our model to estimate $\mathbf{a}_j$ for all 47 annotators in the Presence-1 dataset. (2) For each of 2000 target values (half with $z_i = 1$), we sampled labels from m randomly chosen workers, where the labels were generated according to the estimated $\mathbf{a}_j$ and Equation 7.10. As can be seen from Figure 7.7, our model achieves a consistently lower error rate on synthetic data.

Online algorithm: We simulated running the online algorithm on the Presence datasets obtained using MTurk and used the result from the batch algorithm as

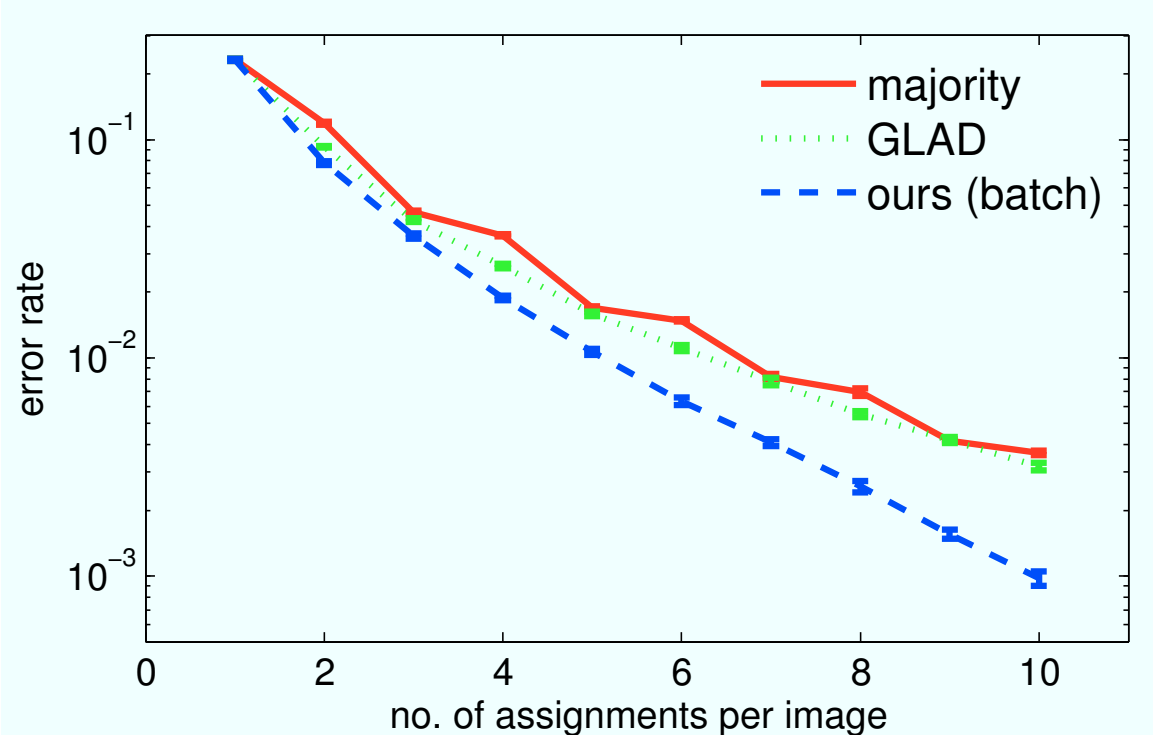

Figure 7.7: Comparison between the majority rule, GLAD [WRW+09], and our algorithm on synthetic data as the number of assignments per image is increased. The synthetic data is generated by the model in Section 7.6 from the worker parameters estimated in Figure 7.5a.

ground truth. When the algorithm requested labels for an image, it was given labels from the dataset (along with an identifier for the worker that provided it) randomly sampled without replacement. If it requested labels from the expert-list for a particular image, it only received such a label if a worker in the expert-list had provided a label for that image, otherwise it was randomly sampled from non bot-listed workers. A typical run of the algorithm on the Presence-1 dataset is shown in Figure 7.9. In the first few iterations, the algorithm is pessimistic about the quality of the annotators, and requests up to $m = 15$ labels per image. As the evidence accumulates, more workers are put in the expert- and bot-lists, and the number of labels requested by the algorithm decreases. Notice in the figure that towards the final iterations, the algorithm samples only 2–3 labels for some images.

To get an idea of the performance of the online algorithm, we compared it to running the batch version from Section 7.4 with limited number of labels per image. For the Presence-1 dataset, the error rate of the online algorithm is almost three times lower than the general algorithm when using the same number of labels per image, see Figure 7.8. For the Presence-2 dataset, twice as many labels per image are needed for the batch algorithm to achieve the same performance as the online version.

It is worth noting that most of the errors made by the online algorithm are on images where the intrinsic uncertainty of the ground truth label is high, i.e. $|R_i|$ as estimated by the full model using all 15 labels per image is large. Indeed, counting errors only for images where $|R_i| > 2$ (using log base 10), which includes 92% of the dataset, makes the error of the online algorithm drop to $0.75\% \pm 0.04\%$ on Presence-1. Thus, the performance clearly depends on the task at hand. If the task is easy, and most annotators agree, it will require few labels per image. If the task is difficult, such that even experts disagree, it will request many labels. The tradeoff between the number of labels requested and the error rate depends on the parameters used. Throughout our experiments, we used $m = 15$, $n = 20$, $\tau' = 2$, $\theta_v = 8 \times 10^{-3}$.

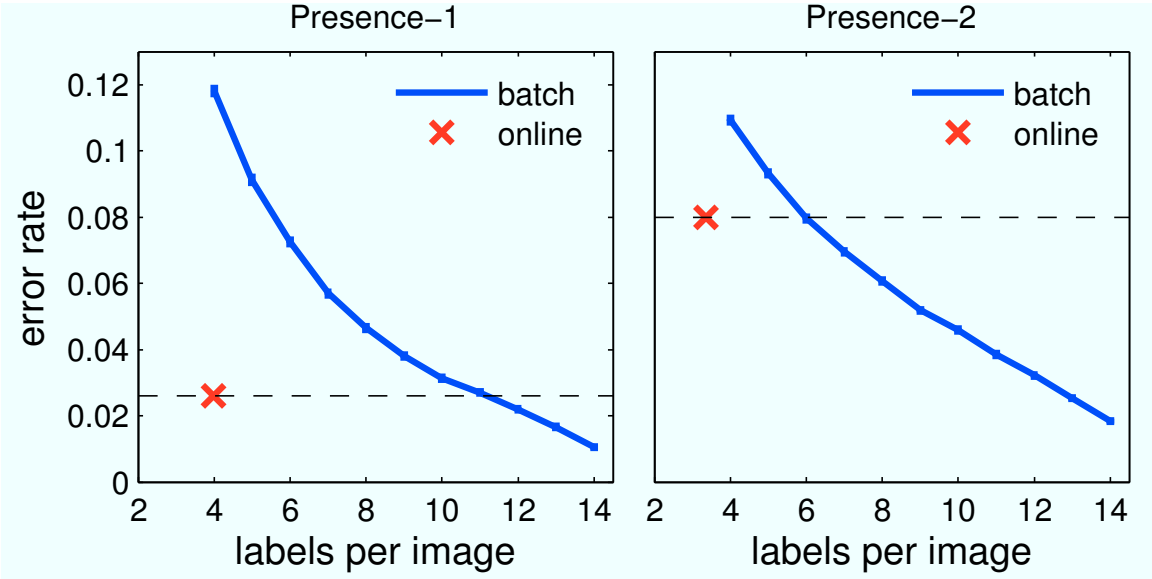

Figure 7.8: Error rates vs. the number of labels used per image on the Presence datasets for the online algorithm and the batch version. The ground truth was the estimates when running the batch algorithm with all 15 labels per image available (thus batch will have zero error at 15 labels per image).

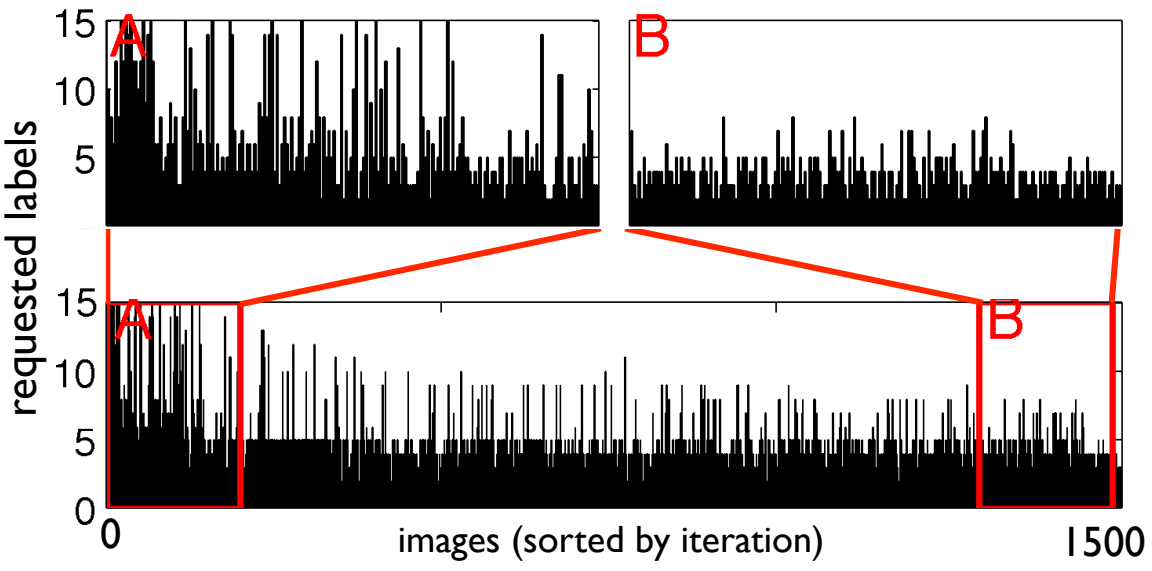

Figure 7.9: Progress of the online algorithm on a random permutation of the Presence-1 dataset. See Section 7.8 for details.

7.9 Conclusions

We have proposed an *online* algorithm to determine the "ground truth value" of some property in an image from multiple noisy annotations. As a by-product it produces an estimate of annotator expertise and reliability. It actively selects which images to label based on the uncertainty of their estimated ground truth values, and the desired level of confidence. We have shown how the algorithm can be applied to different types of annotations commonly used in computer vision: binary yes/no annotations, multi-valued attributes, and continuous-valued annotations (e.g. bounding boxes).

Our experiments on MTurk show that the quality of annotators varies widely in a continuum from highly skilled to almost random. We also find that equally skilled annotators differ in the relative cost they attribute to false alarm errors and to false reject errors. Our algorithm can estimate this quantity as well.

Our algorithm minimizes the labeling cost by assigning the labeling tasks preferentially to the best annotators. By combining just the right number of (possibly noisy) labels it defines an optimal 'virtual annotator' that integrates the real annotators without wasting resources. Thresholds in this virtual annotator may be designed optimally to trade off the cost of obtaining one more annotation with the cost of false alarms and the cost of false rejects. Future work includes dynamic adjustment of the price paid per annotation to reward high quality annotations and to influence the internal thresholds of the annotators.

Chapter 8

Performance Estimation

8.1 Abstract

How many labeled examples are needed to estimate a classifier's performance on a new dataset? We study the case where data is plentiful, but labels are expensive. We show that by making a few reasonable assumptions on the structure of the data, it is possible to estimate performance curves, with confidence bounds, using a small number of ground truth labels. Our approach, which we call Semisupervised Performance Evaluation (SPE), is based on a generative model for the classifier's confidence scores, and a novel active sampling scheme. SPE can be used to recalibrate a classifier that is being applied to new data by re-estimating the class-conditional confidence distributions. We show how SPE with active sampling outperforms other approaches in terms of labels required to estimate classifier performance.

8.2 Introduction

Consider a biologist who downloads software for classifying the behavior of fruit flies. The classifier was laboriously trained by a different research group who labeled thousands of training examples to achieve satisfactory performance on a validation set collected in some particular setting (see e.g. [DWH+09]). The biologist would be ill-advised if she trusted the published performance figures; maybe small lighting changes in her experimental setting have changed the statistics of the data and rendered the

classifier useless. However, if the biologist has to go over all the labels assigned by the classifier to her dataset, just to be sure the classifier is performing up to expectation, then what is the point in obtaining a trained classifier in the first place? Is it possible at all to obtain a reliable evaluation of a classifier when unlabeled data is plentiful, but when the user is willing to provide only a small number of labeled examples?

We propose a method for achieving minimally supervised evaluation of classifiers, requiring as few as 10 labels to accurately estimate classifier performance. Our method is based on a generative Bayesian model for the confidence scores produced by the classifier, borrowing from the semi supervised literature [NMTM00, See02, Zhu08], and on an active strategy for obtaining the most informative labels. An additional contribution is a fast approximate inference method for doing inference in our model.

8.3 Modeling the Classifier Score

Let us start with a set of N data items, $(x_i, y_i) \in \mathcal{R}^D \times \{0, 1\}$, drawn from some unknown distribution $p(x, y)$ and indexed by $i \in \{1, \ldots, N\}$. Suppose that a classifier, $\bar{h}(x_i; \tau) = [h(x_i) > \tau]$, where τ is some scalar threshold, has been used to classify all data items into two classes, $\hat{y}_i \in \{0, 1\}$. While the "ground truth" labels y_i are assumed to be unknown, initially, we do have access to all the "scores," $s_i = h(x_i)$, computed by the classifier. From this point onwards, we forget about the data vectors x_i and concentrate solely on the scores and labels, $(s_i, y_i) \in \mathcal{R} \times \{0, 1\}$.

The key assumption in this paper is that the list of scores $S = (s_1, \ldots, s_N)$ and the unknown labels $Y = (y_1, \ldots, y_N)$ can be modeled by a two-component mixture model $p(S, Y \mid \theta)$, parameterized by θ, where the class-conditionals are standard parametric distributions. We show in Section 8.6.2 that this is a reasonable assumption for many datasets.

Suppose that we can ask an expert (the "oracle") to provide the true label y_i for any data item. This is an expensive operation and our goal is to ask the oracle for as few labels as possible. The set of items that have been labeled by the oracle at time t is denoted by $\mathcal{L}_t$ and its complement, the set of items for which the ground truth

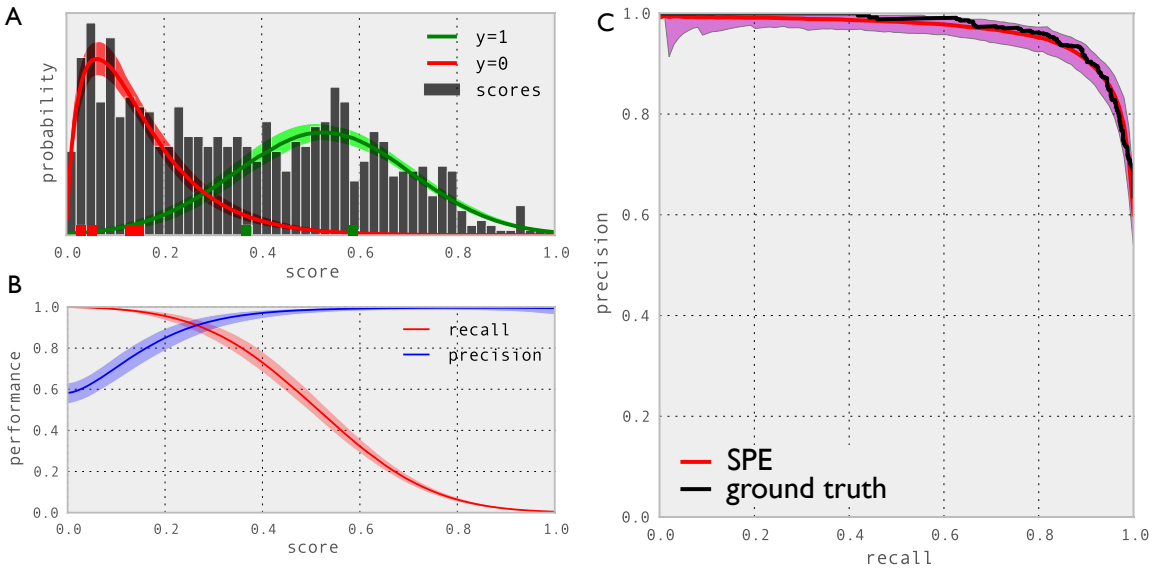

Figure 8.1: Estimating detector performance with all but 10 labels unknown. **A**: Histogram of classifier scores s_i obtained by running the "ChnFtrs" detector [DWSP11] on the INRIA dataset [DT05]. The red and green curves show the Gamma-Normal mixture model with highest likelihood. The shaded bands indicate the 90% probability bands around the model. The red and green bars show the labels of the 10 randomly sampled labels (active querying was not used). **B**: Precision and recall curves computed from the mixture model in A. **C**: In black, precision-recall curve computed after all items have been labeled. In red, precision-recall curve estimated using SPE from only 10 labeled examples (with 90% confidence interval shown as the magenta band). See Section 8.3 for a discussion.

is unknown, is denoted $\mathcal{U}_t$. This setting is similar to semisupervised learning [See02, Zhu08]. By estimating $p(S, Y \mid \theta)$, we will improve our estimate of the performance of $\bar{h}$ when $|\mathcal{L}_t| \ll N$.

Consider first the fully supervised case, i.e. where all labels y_i are known. Let the scores s_i be i.i.d. according to the two mixture model. If the all labels are known, the likelihood of the data is given by,

$$p(S, Y \mid \theta) = \prod_{i:y_i=0} (1-\pi) p_0(s_i \mid \theta_0) \prod_{i:y_i=1} \pi p_1(s_i \mid \theta_1), \tag{8.1}$$

where $\theta = \{\pi, \theta_0, \theta_1\}$, and $\pi \in [0, 1]$ is the mixture weight, i.e. $p(y_i = 1) = \pi$. The component densities p_0 and p_1 could be Normal distributions, Gamma distributions, or some other probability distributions appropriate for the given classifier (see Section 8.6.2 for a discussion about which class conditional distributions to choose). This approach of applying a generative model to score distributions, when all labels are known, has been used in the past to obtain error estimates on classifier performance [HAS99, ESCA06, GGR08], and for classifier calibration [Ben02]. However, previous approaches require that the all items used to estimate the performance have been labeled.

One contribution of this work is the realization that it is possible to estimate classifier performance even when only a fraction of the ground truth labels are known. In this case, the labels for the unlabeled items $i \in \mathcal{U}_t$ can be marginalized out,

$$\begin{aligned} p(S, Y_t \mid \theta) = & \prod_{i \in \mathcal{U}_t} \left((1-\pi) p_0(s_i \mid \theta_0) + \pi p_1(s_i \mid \theta_1) \right) \\ & \times \prod_{i \in \mathcal{L}_t} \pi^{y_i} (1-\pi)^{1-y_i} p_{y_i}(s_i \mid \theta_{y_i}), \end{aligned} \tag{8.2}$$

where $Y_t = \{y_i \mid i \in \mathcal{L}_t\}$. This allows the model to make use of the scores of unlabeled items in addition to the labeled items, which enables accurate performance estimates with only a handful of labels. Once we have the likelihood, we can take a Bayesian approach to estimate the parameters θ. Starting from a prior on the parameters, $p(\theta)$,

we can obtain a posterior $p(\theta \mid S, Y_t)$ by using Bayes' rule,

$$p(\theta \mid S, Y_t) \propto p(S, Y_t \mid \theta)\, p(\theta). \tag{8.3}$$

Let us look at a real example. Figure 8.1a shows a histogram of the scores obtained from classifier on a public dataset (see Section 8.6 for more information about the datasets we use). At first glance, it is difficult to guess the performance of the classifier unless the oracle provides a lot of labels. However, if we *assume* that the scores follow a two-component mixture model as in (8.2), with a Gamma distribution for the $y_i = 0$ and a Normal distribution for the $y_i = 1$ component, then there is a only a narrow choice of θ that can explain the scores with high likelihood; the red and green curves in Figure 8.1a show such a high probability hypothesis. As we will see in the next section, the posterior on θ can be used to estimate the performance of the classifier.

8.4 Estimating Performance

Most performance measures can be computed directly from the model parameters θ. For example, two often used performance measures are the precision $P(\tau; \theta)$ and recall $R(\tau; \theta)$ at a particular score threshold τ. We can define these quantities in terms of the conditional distributions $p_y(s_i \mid \theta_y)$. Recall is defined as the fraction of the positive, i.e. $y_i = 1$, examples that have scores above a given threshold,

$$R(\tau; \theta) = \int_{\tau}^{\infty} p_1(s \mid \theta_1)\, ds. \tag{8.4}$$

Precision is defined to be the fraction of all examples with scores above a given threshold that are positive,

$$P(\tau; \theta) = \frac{\pi R(\tau; \theta)}{\pi R(\tau; \theta) + (1 - \pi) \int_{\tau}^{\infty} p_0(s \mid \theta_0)\, ds}. \tag{8.5}$$

We can also compute the precision at a given level of recall by inverting $R(\tau; \theta)$, i.e. $P_r(r; \theta) = P(R^{-1}(r; \theta); \theta)$ for some recall r. Other performance measures, such as the

equal error rate, true positive rate, true negative rate, sensitivity, specificity, and the ROC can be computed from θ in a similar manner.

The posterior on θ can also be used to obtain confidence bounds on the performance of the classifier. For example, for some choice of parameters θ, the precision and recall can be computed for a range of score thresholds τ to obtain a curve (see solid curves in Figure 8.1b). Similarly, given the posterior on θ, the distribution of $P(\tau;\theta)$ and $R(\tau;\theta)$ can be computed for a fixed τ to obtain confidence intervals (shown as colored bands in Figure 8.1b). The same reasoning can be applied to the precision-recall curve: for some recall r, the distribution of precisions, found using $P_r(r;\theta)$ can be used to compute confidence intervals on the curve (see Figure 8.1c).

While the approach of estimating performance based purely on the estimate of θ works well in limit when $N \to \infty$, it has some drawbacks when N is small (on the order of 10^3–10^4) and π is unbalanced, in which case finite-sample effects come into play. Since it views the scores (and the associated labels) as a finite sample from $p(S, Y \mid \theta)$, there will always be uncertainty in the performance estimate. When all items have been labeled by the oracle, the remaining uncertainty in the performance represents the variability in sampling (S, Y) from $p(S, Y \mid \theta)$. In practice, however, the question that is often asked is, "What is our best guess for the classifier performance on this particular test set?" In other words, we are interested in the sample performance rather than the population performance. Thus, when the oracle has labeled the whole test set, there should not be any uncertainty in the performance; it can simply be computed directly from (S, Y).

To estimate the sample performance, we need to account for uncertainty in the unlabeled items, $i \in \mathcal{U}_t$. This uncertainty is captured by the distribution of the unobserved labels $Y'_t = \{y_i \mid i \in \mathcal{U}_t\}$, found by marginalizing out the model parameters,

$$\begin{aligned} p(Y'_t \mid S, Y_t) &= \int_\Theta p(Y'_t, \theta \mid S, Y_t)\, d\theta \\ &= \int_\Theta p(Y'_t \mid \theta) p(\theta \mid S, Y_t)\, d\theta. \end{aligned} \tag{8.6}$$

Here Θ is the space of all possible parameters. On the second line we rely on the

assumption of a mixture model to factor the joint probability distribution on θ and Y_t'.

One way to think of this approach is as follows: imagine that we sample Y_t' from $p(Y_t' \mid S, Y_t)$. We can then use all the labels $Y = Y_t \cup Y_t'$ and the scores S to trace out a performance curve (e.g., a precision-recall curve). Now, as we repeat the sampling, each performance curve will look slightly different. Thus, the posterior distribution on Y_t' in effect gives us a distribution of performance curves. We can use this distribution to compute quantities such as the expected performance curve, the variance in the curves, and confidence intervals. The main difference between the sample and population performance estimates will be at the tails of the score distribution, $p(S \mid \theta)$, where individual item labels can have a large impact on the performance curve.

8.4.1 Sampling from the posterior

In practice, we cannot compute $p(Y_t' \mid S, Y_t)$ in (8.6) analytically, so we must resort to approximate methods. For some choices of class conditional densities, $p_y(s \mid \theta_0)$, such as when they are Normal distributions, it is possible to carry out the marginalization over θ in (8.6) analytically. In that case one could use collapsed Gibbs sampling to sample from the posterior on Y_t', as is often done for models involving the Dirichlet process [Mac94]. A more generally applicable method, which we will describe here, is to split the sampling into three steps: (a) sample $\bar{\theta}$ from $p(\theta \mid S, Y_t)$, (b) fix the mixture parameters to $\bar{\theta}$ and sample the labels Y_t' given their associated scores, and (c) compute the performance, such as precision and recall, for all score thresholds $\tau \in S$. By repeating these three steps, we can obtain a sample from the distribution over the performance curves.

The first step, sampling from the posterior $p(\theta \mid S, Y_t)$, can be carried out using importance sampling (IS). We experimented with Metropolis-Hastings and Hamiltonian Monte Carlo [Nea10], but we found that IS worked well for this problem, required less parameter tuning, and was much faster. In IS, we sample from a proposal distribution

$q(\theta)$ in order to estimate properties of the desired distribution $p(\theta \mid S, Y_t)$. Suppose we draw M samples of θ from $q(\theta)$ to get $\bar{\Theta} = \{\bar{\theta}^1, \ldots, \bar{\theta}^M\}$. Then, we can approximate expectations of some function $g(\cdot)$ using the weighted function evaluations, i.e. $E[g] \simeq \sum_{m=1}^{M} w_m g(\bar{\theta}^m)$. The weights $w_m \in W$ correct for the bias introduced by sampling from $q(\theta)$ and are defined as,

$$w_m = \frac{p(\bar{\theta}^m \mid S, Y_t)/q(\bar{\theta}^m)}{\sum_l p(\bar{\theta}^l \mid S, Y_t)/q(\bar{\theta}^l)}. \tag{8.7}$$

For the datasets in this paper, we found that the state-space around the MAP estimate[1] of θ,

$$\theta^\star = \arg\max_\theta p(\theta \mid S, Y_l), \tag{8.8}$$

was well approximated by a multivariate Normal distribution. Hence, for the proposal distribution we used,

$$q(\theta) = \mathcal{N}(\theta \mid \mu_q, \Sigma_q). \tag{8.9}$$

To simplify things further, we used a diagonal covariance matrix, Σ_q. The elements along the diagonal of Σ_q were found by fitting a univariate Normal locally to $p(\theta \mid S, Y_t)$ along each dimension of θ while the other elements were fixed at their MAP-estimates. The mean of the proposal distribution, μ_q, was set to the MAP estimate of θ.

We now have all steps needed to estimate the performance of the classifier, given the scores S and some labels Y_t obtained from the oracle:

1. Find the MAP estimate μ_q of θ using (8.8).
2. Fit a proposal distribution $q(\theta)$ to $p(\theta \mid S, Y_t)$ locally around μ_q.
3. Sample M instances of θ, $\bar{\Theta} = \{\bar{\theta}^1, \ldots, \bar{\theta}^M\}$, from $q(\theta)$ and calculate the weights $w_m \in W$.
4. For each $\bar{\theta}^m \in \bar{\Theta}$, sample the labels for $i \in \mathcal{U}_t$ to get $\bar{Y}'_t = \{\bar{Y}'_{t,1}, \ldots, \bar{Y}'_{t,M}\}$.

[1] We used BFGS-B [BLNZ95] to carry out the optimization. To avoid local maximums, we used multiple starting points.

5. Estimate performance measures using the scores S, labels $\bar{Y}_{t,m} = Y_t \cup \bar{Y}'_{t,m}$ and weights $w_m \in W$.

8.5 Querying the Oracle

So far we have assumed that $\mathcal{L}_t$ is fixed. Is it possible to emulate active learning and have the oracle label the most informative items? In this section we present a strategy for "active querying," where the items to be labeled are chosen to reduce the uncertainty in the classifier performance the most. Thus, our approach is related to active learning approaches [Mac92, CGJ96, DH08] where labels are sampled to decrease the uncertainty in model parameters.

Suppose that we have some performance measure $f(Y'_t; Y_t, S)$ that is a function of the known and unknown labels, Y_t and Y'_t, and scores S at time t. For example, f may be a precision recall curve, an ROC, etc. To keep the notation uncluttered, we will denote $f(Y'_t) \equiv f(Y'_t; Y_t, S)$. Since Y'_t is a random variable, so is $f(Y'_t)$. Suppose further that we have some measure $U(\cdot)$ of "uncertainty" so that $U(f(Y'_t))$ is the uncertainty in $f(Y'_t)$. For example, as we shall see below, the uncertainty may be related to the variance in the performance due to the resampling variance of Y'_t.

What item should the oracle label next? The strategy we adopt is to label the item for which the expected uncertainty in $f(Y'_t)$ decreases the most, i.e.,

$$k_t = \arg\max_{k \in \mathcal{U}_t} U(f(Y'_t)) - E\left[U(f(Y'_t \setminus \{y_k\}))\right], \tag{8.10}$$

where $E[\cdot]$ denotes the expectation over y_k. We call this querying strategy "active querying"; at every point in time t, the oracle is asked to label the item k_t that maximizes the reduction in uncertainty.

There are many choices for f and U, and which ones to choose depend on the application. Here, we will use the precision-recall curve as the performance measure f. That is, we define $f(Y'_t) \equiv P(r; Y'_t) \mid r \in [0, 1]$, where $P(r; Y'_t)$ is the precision at recall r given the unknown labels Y'_t. The measure of uncertainty also depends on

the application. Here, we are interested in reducing the uncertainty in our estimate of the performance curve, so we define it to be the area of the confidence interval of the performance curve,

$$U(f(Y_t')) = \int_0^1 \sqrt{\mathrm{Var}\left[P(r; Y_t')\right]}\, dr, \tag{8.11}$$

where,

$$\mathrm{Var}\left[P(r; Y_t')\right] = E\left[P(r; Y_t')^2\right] - E\left[P(r; Y_t')\right]^2. \tag{8.12}$$

The expectation in (8.12) is over Y_t'.

Realistically, carrying out the maximization in (8.10) and the sampling required to compute (8.11) can be quite computation intensive. To make the approach more practical, we make the following approximation: Instead of carrying out the maximization of (8.10) over all $k \in \mathcal{U}_t$, we maximize over a subset $\mathcal{U}_t' \subset \mathcal{U}_t$ with $|\mathcal{U}_t'| \ll |\mathcal{U}_t|$. The items in $\mathcal{U}_t'$ are chosen so that the items' scores s_i are roughly equally spaced over the range of all scores S, under the assumption that labeling items with similar scores will have roughly the same impact on U.[2]

Note that by using the active querying strategy presented here, our estimate of the performance curve will no longer be unbiased. This is because there will be correlations in the labels queried by the active strategy. One situation where this may cause problems is if the class conditionals are not well-explained by standard parametric distributions (see Figure 8.2c for an example), in which case the model may get stuck in a local optimum. While the problem of unbiased active sampling is an ongoing research topic [DH08, BDL09], biased sampling often works well in practice [TK01]. Similarly, we did not observe any issues in our experiments.

[2]We tried different sizes of $\mathcal{U}_t'$ between 10 and 30 with similar results.

8.6 Experiments

8.6.1 Datasets

We surveyed the literature for published classifier scores with ground truth labels. One such dataset that we found was the Caltech Pedestrian Dataset[3] (CPD), for which both detector scores and ground truth labels are available for a wide variety of detectors [DWSP11]. Moreover, the CPD website also has scores and labels available, using the same detectors, for other pedestrian detection datasets, such as the INRIA (abbr. INR) dataset [DT05].

We made use of the detections in the CPD and INR datasets as if they were classifier outputs. To some extent, these detectors are in fact classifiers, in that they use the sliding-window technique for object detection. Here, windows are extracted at different locations and scales in the image, and each window is classified using a pedestrian classifier (with the caveat that there is often some extra post-processing steps carried out, such as non-maximum suppression to reduce the number of false positive detections). For our experiments, we concentrated on two of the best performing detectors, "ChnFtrs" and "LatSvm-V2," run on the CPD and INR test sets, but similar results were obtained for the other detectors. To make experiments go faster, we sampled the datasets randomly to have between 800–2,000 items. See [DWSP11] for references to all detectors.

To complement the pedestrian datasets, we also used a basic linear SVM classifier and a logistic regression classifier on the "optdigits" (abbr. DGT) and "sat" (SAT) datasets from the UCI Machine Learning Repository [FA10]. Since both datasets are multiclass, but our method only handles binary classification, we chose one category for $y = 1$ and grouped the others into $y = 0$. Planned future work includes extending our approach to multiclass classifiers. In the figures, "svm3" is used to mean that the SVM classifier was used with category 3 in the dataset being assigned to the $y = 1$ class, and "logres9" denotes that the logistic regression classifier was used with

[3]Downloaded from `http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/`.

category 9 being the $y = 1$ class. The datasets had 1,800–2,000 items each.

8.6.2 Choosing class conditionals

So far we have not discussed in detail which distribution families to use for the class conditional $p_y(s \mid \theta_y)$ distributions. To find out which parametric distributions are appropriate for modeling the score class-conditionals, we took the classifier scores and split them into two groups, one for $y_i = 0$ and one for $y_i = 1$. We used MLE to fit different families of probability distributions (see Table 8.1 for a list of distributions) on 80% of the data (sampled randomly) in each group. We then ranked the distributions by the log likelihood of the remaining 20% of the data (given the MLE-fitted parameters). In total, we carried out this procedure on 78 class conditionals from the different datasets and classifiers.

Table 8.1 shows the top-3 distributions that explained the class-conditional scores with highest likelihood for a selection of the datasets and classifiers. We found that the truncated Normal distribution was in the top-3 list for 48/78 dataset class-conditionals, and that the Gamma distribution was in the top-3 list 53/78 times; at least one of the two distributions were always in the top-3 list. Figure 8.2 show some examples of the fitted distributions from Table 8.1. In some cases, like Figure 8.2c, a mixture model would have provided a better fit than the simple distributions we tried. That said, we found that truncated Normal and Gamma distributions were good choices for most of the datasets.

Since we use a Bayesian approach in equation (8.3), we must also define a prior on θ. The prior will vary depending on which distribution is chosen, and it should be chosen based on what we know about the data and the classifier. As an example, for the truncated Normal distribution, we use a Normal and a Gamma distribution as priors on the mean and standard deviation, respectively (since we use sampling for inference, we are not limited to conjugate priors). As a prior on the mixture weight π, we use a Beta distribution.

In some situations when little is known about the classifier, it makes sense to

dataset	1st	2nd	r.l.l.	3rd	r.l.l.
(CPD) ChnFtrs [0]	g	ln	1.00	f-r	0.99
(CPD) ChnFtrs [1]	f-r	n	0.99	gu-r	0.97
(CPD) LatSvmV2 [0]	ln	g	1.00	f-r	0.99
(CPD) LatSvmV2 [1]	f-r	g	0.97	gu-r	0.96
(CPD) FeatSynth [0]	n	f-r	0.99	g	0.98
(CPD) FeatSynth [1]	n	g	1.00	f-r	0.98
(INR) LatSvmV2 [0]	ln	g	0.99	f-r	0.98
(INR) LatSvmV2 [1]	n	f-r	0.87	gu-l	0.73
(INR) ChnFtrs [0]	g	f-r	1.00	n	0.97
(INR) ChnFtrs [1]	f-r	n	0.97	g	0.84
(DGT) logres9 [0]	f-r	n	0.98	gu-l	0.96
(DGT) logres9 [1]	f-r	gu-l	0.99	n	0.99
(SAT) svm3 [0]	n	f-r	0.98	gu-r	0.84
(SAT) svm3 [1]	n	g	0.99	ln	0.98

Table 8.1: Distributions best representing empirical class-conditional score distributions (for a subset of the 78 cases we tried). Each row shows the top-3 distributions, i.e. explaining the class-conditional scores with highest likelihood, for different combinations of datasets, classifiers and the class-labels (shown in brackets, $y = 0$ or $y = 1$). The distribution families we tried included (with abbreviations used in last three columns in parentheses) the truncated Normal (n), truncated Student's t (t), Gamma (g), log-normal (ln), left- and right-skewed Gumbel (g-l and g-r), Gompertz (gz), and Frechet right (f-r) distribution. The last and second to last column show the relative log likelihood (r.l.l.) with respect to the best (1st) distribution. Figure 8.2 shows examples of fitted distributions.

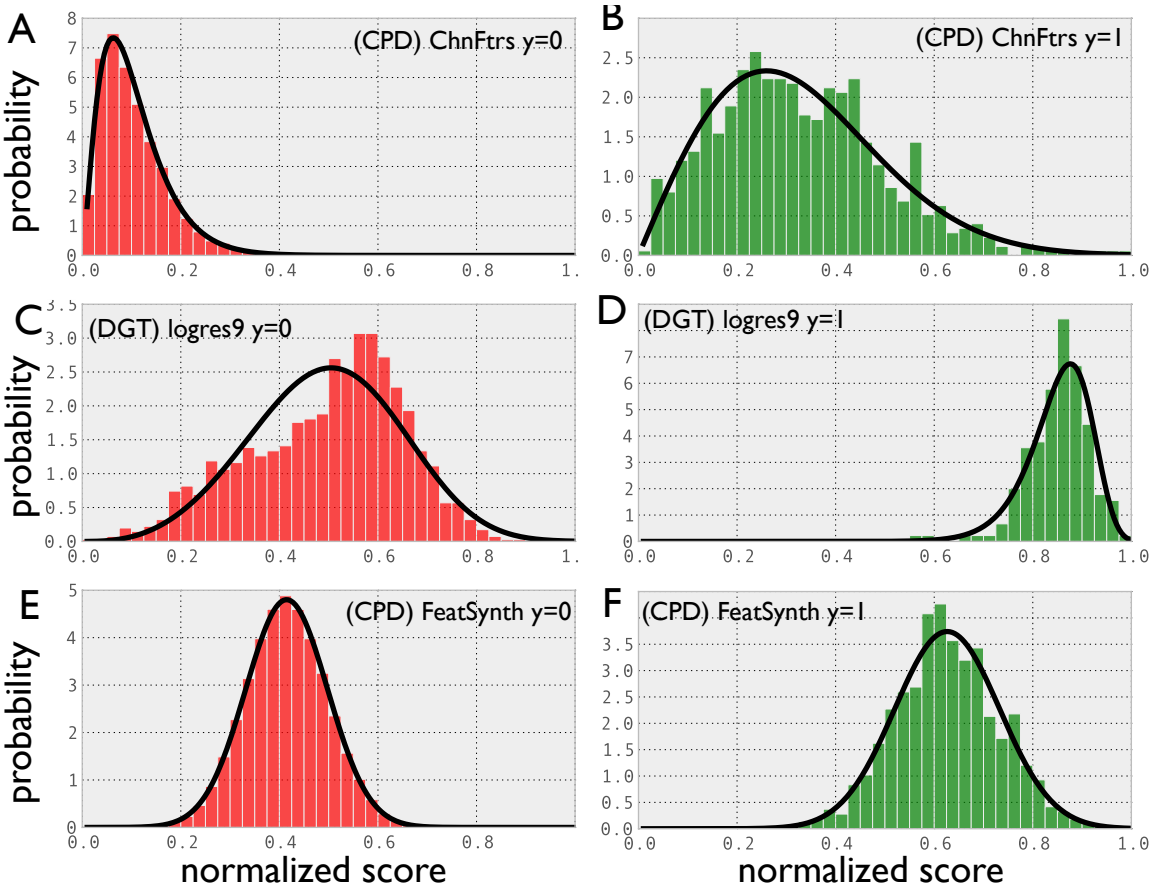

Figure 8.2: Standard parametric distributions $p_y(s \mid \theta_y)$ (black solid curve) fitted to the class conditional scores for a few example datasets and classifiers. The score distributions are shown as histograms. In all cases, we normalized the scores to be in the interval $s_i \in (0, 1]$, and made the truncation at $s = 0$ for the truncated distributions. See Section 8.6.2 for more information.

try different kinds of class-conditional distributions. One heuristic, which we found worked well in our experiments, is to try different combinations of distributions for p_0 and p_1, and then choose the combination achieving the highest maximum likelihood on the labeled and unlabeled data.

8.6.3 Comparing querying schemes

We compared the active querying scheme to random querying, as described in Section 8.5. Shown in Figure 8.3 is the error in the estimated performance, measured as the area between the estimated precision-recall curve and the ground truth curve, for four different datasets. In all cases, SPE outperforms the standard approach of computing the performance on only the labeled items (green curve). Figure 8.4 shows the estimated performance curves using SPE with random and active querying.

The main advantage of the active querying scheme is when the class conditionals are unbalanced, in which case it can converge faster than random querying (see Figure 8.3d). This is often the case for the CPD datasets, where the detectors produce a lot of false negatives, leaving only between 0.1–6% items from $y = 1$. In such situations, the $y = 0$ items dominate the score distribution. Thus, random querying will reveal mostly $y_i = 0$ labels, and not much is learned about the shape of the p_1 class-conditional. However, in those situations, active sampling keeps requesting labels for items with high scores. This is because $y = 0$ items with high scores can create large fluctuations in the low-recall, high-precision region of the performance curve, so knowing those labels reduces the uncertainty in the curve a lot (see Figure 8.4b, where only 0.5% of the items were from the $y = 1$ class). Since, for a good classifier, the $y = 1$ items have higher scores on average, this allows the active sampling to get a better estimate of the shape of the p_1 class-conditional, while still using a majority of the unlabeled data to estimate the shape of p_0.

When the class-conditionals are balanced, i.e. $\pi \simeq 0.5$, the two sampling methods worked equally effectively. In that case, the scores for the items queried by the active strategy are distributed more uniformly. In some situations, for example, the ChnFtrs

classifier on the INR dataset, either method required only 10 labels to estimate the performance curve with high accuracy (see Figure 8.1). Thus, in those cases, sampling more than 10 labels does not provide any advantage, other than that the confidence intervals shrink.

8.6.4 Classifier recalibration

Applying SPE to a test dataset allows us to "recalibrate" the classifier to that dataset. Unlike previous work on classifier calibration [Ben02, Pla99], SPE does not require all items to be labeled. For each unlabeled data item, we can compute the probability that it belongs to the $y = 1$ class by calculating the empirical expectation from the samples, i.e. $\hat{p}(y_i = 1) = E[y_i = 1 \mid S, Y_t]$.

Similarly, we can choose a threshold τ to use with the classifier $\bar{h}(x_i; \tau)$ based on some pre-determined criteria. For example, the requirement might be that the classifier performs with recall $R(\tau) > \hat{r}$ and precision $P(\tau) > \hat{p}$. In that case, we define a condition $C(\tau) = [R(\tau) > \hat{r} \wedge P(\tau) > \hat{p}]$. Then, for each τ, we find the probability that the condition is satisfied by calculating the expectation $\hat{p}(C(\tau) = 1) = E[C(\tau)]$ over the unlabeled items Y'_t. Figure 8.5 shows the probability that $C(\tau)$ is satisfied at different values of τ. Thus, this approach can be used to choose new thresholds for different datasets.

8.7 Discussion

We explored the problem of estimating classifier performance from few labeled items. Using four public datasets, and multiple classifiers, we showed that classifier score distributions can often be well approximated by two-component mixture models with standard parametric component distributions. Borrowing ideas from semisupervised learning, we demonstrated how our model, Semisupervised Performance Evaluation (SPE), can be used to estimate classifier performance, with confidence intervals, using only a few labeled examples. We presented a sampling scheme based on importance sampling for efficient inference. Furthermore, we showed a novel active querying

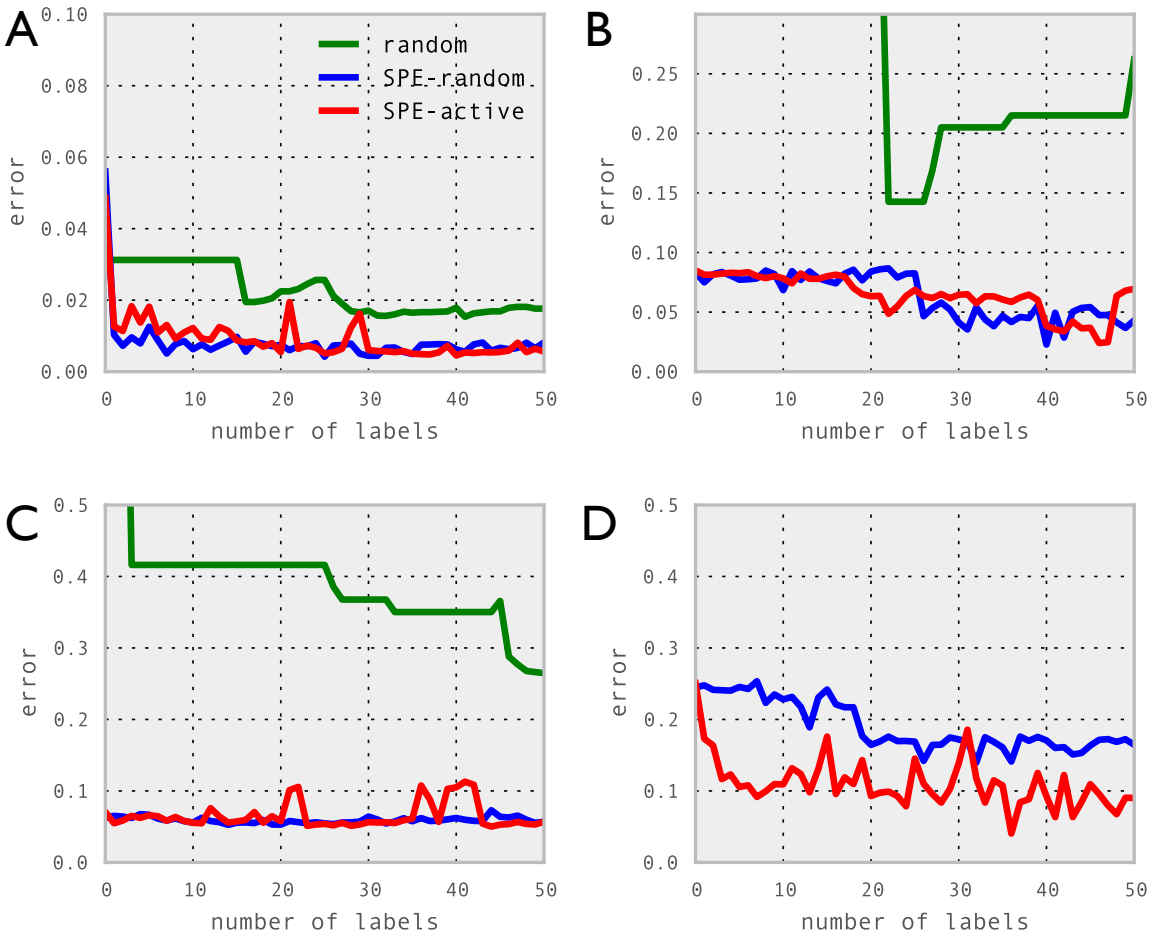

Figure 8.3: SPE with active and random label querying versus number of labels queried. The plots show the error, measured as the area between the estimated precision-recall curves and the ground truth curve, as the number of labels queried from the oracle increases. The green curve (random) shows the performance estimated based only on the queried labels without SPE. The red and blue curves show the result of using SPE (and thus taking advantage of the unlabeled data) with the active and random querying schemes. **A:** INR using ChnFtrs. **B:** INR using C. **C:** DGT with svm8. **D:** CPD using LatSvmV2 (green curve is outside axis bounds). See Section 8.6.3 for a discussion.

strategy for obtaining labels that is more advantageous than random sampling when the classes are unbalanced.

This line of research opens up many interesting avenues for future exploration. For example, is it possible to do unbiased active querying? One possibility in this direction would be to employ importance weighted active sampling techniques [BDL09, DH08]. Another future direction would be to extend SPE to multi-component mixture models and multiclass problems. That said, as shown by our experiments, SPE already works well for a broad range of classifiers and datasets, and can estimate classifier performance with as few as 10 labels.

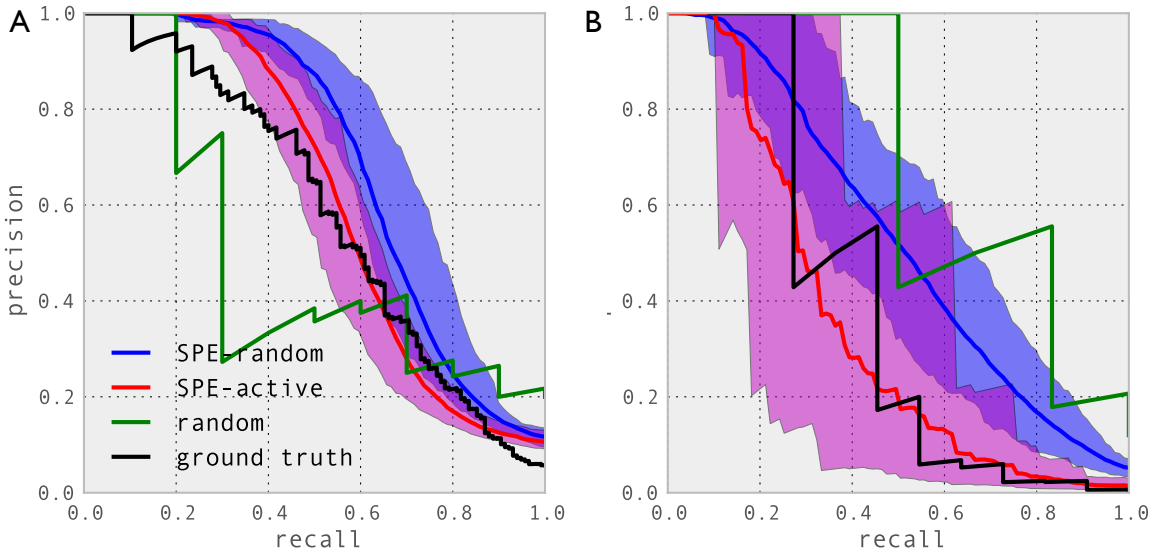

Figure 8.4: Estimated precision and recall after 50 labels have been queried. The blue and red curve shows the result of using SPE with random and active querying scheme (with 90% confidence bands). The green curve shows the precision and recall computed only using the 50 labels (no SPE), and the black curve shows the ground truth. **A:** CPD using ChnFtrs. **B:** CPD using LatSvmV2. See Section 8.6.3 for a discussion.

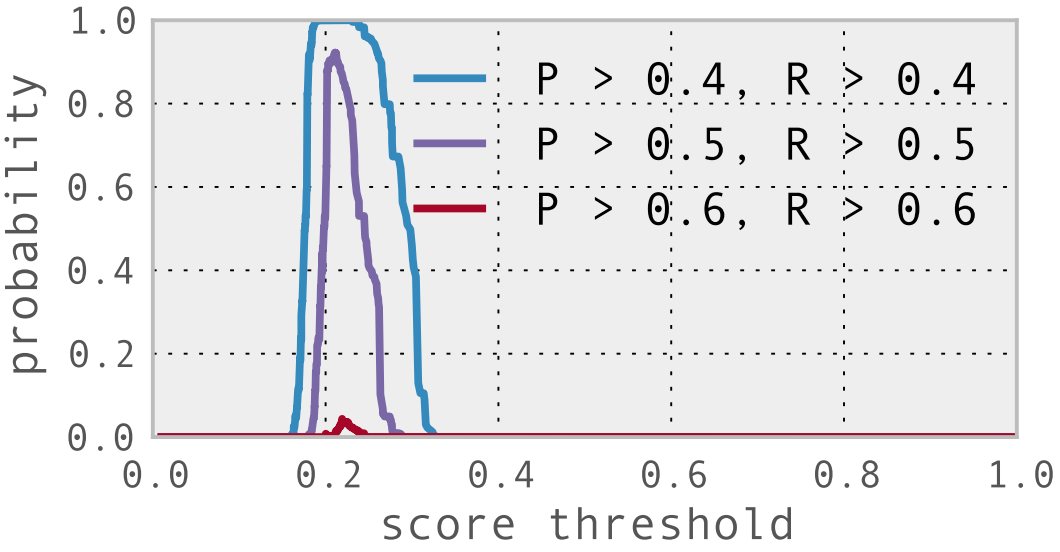

Figure 8.5: Recalibrating the classifier by estimating the probability that a condition is met. In this case, the three curves show the estimated probability that conditions on precision P and recall R have been met for different score thresholds τ. The data used was the same as in Figure 8.4a. This analysis suggests that the best threshold is $\tau \approx 2.2$.

Chapter 9

Conclusions

The aim of this thesis was to provide the foundation for building Hybrid Human-Machine Vision Systems (HHMVS), where humans and machines work together to annotate images. Today's machine vision systems are not accurate enough for many image annotation tasks. On the other hand, using experts or crowdsourcing is often too expensive for tasks involving large amounts of image data. The goal of HHMVS is to bridge this gap by using machines to annotate most of the data, and let humans annotate and correct labels for images where the machine is uncertain. To that end, we need methods for estimating the accuracy of the machine vision systems with as few expert-provided labels as possible. Moreover, we need to understand how to use crowdsourcing as cost-effectively as possible while still maintaining high quality annotations. In this thesis, we have begun to answer some of the more fundamental questions towards this goal.

The first question we investigated was how annotators perceive images and make decision in binary annotation tasks (see Chapter 3). By modeling annotators as making linear decisions in a multidimensional space, we showed how annotators are often biased to look for slightly different things, and that it is possible to identify experts in a crowd of annotators. Exploiting a simple parameterization of annotator characteristics, including competence, bias and expertise, we further showed how to use crowdsourcing to obtain high quality annotations even with no ground truth labels present. Extensions of the model showed that more informative labels can be obtained using a 3–5 level confidence scale instead of binary questions (see Chapter 4). By

extending the model to pairwise models, it can also be used for "crowd clustering," where a crowd of annotators work together to partition a dataset of images into separate categories (see Chapter 5).

We investigated how crowdsourcing can be used to obtain high quality detection annotations (see Chapter 6). Experiments on real and synthetic images showed that simple voting-based methods can produce good results. We also found that using multiple non-expert annotators can provide an accuracy that is close to that of experts.

We studied the problem of scaling crowdsourcing to very large numbers of images (see Chapter 7). When are we sure of an image label? How quickly can we determine that an annotator is providing low quality work? We proposed a general model for characterizing annotator competence that is applicable to binary, discrete and continuous annotations. By using the model in an online setting, where annotators are rejected if they do not perform well, we showed that the number of labels per image can be greatly reduced while maintaining a high level of accuracy.

Finally, we looked at how to estimate the performance of a classifier with as few expert-provided labels as possible (see Chapter 8). By taking a semi-supervised approach and modeling classifier scores as a mixture model, we showed that the performance can be estimated with as few as 10 labels, provided the classifier satisfies the assumptions of the model.

Each chapter of this thesis has provided a piece in the foundation of HHMVS, but there is much interesting work left before we can build a reliable system. One of the most pressing questions is "How do we combine crowds and machines, now that we have representations of the performance of both?" One possible path is to extend the models presented in Chapters 3 & 7 to cover scores provided by classifiers (Chapter 8). Another interesting path for future work is to combine classifier performance estimation with active learning (see Figure 9.1). The goal would be to stop training the classifier as soon as it reaches high enough accuracy.

HHMVS has the potential to truly revolutionize how science and industry work with images. Instead of considering images to be digital "dark matter," that is,

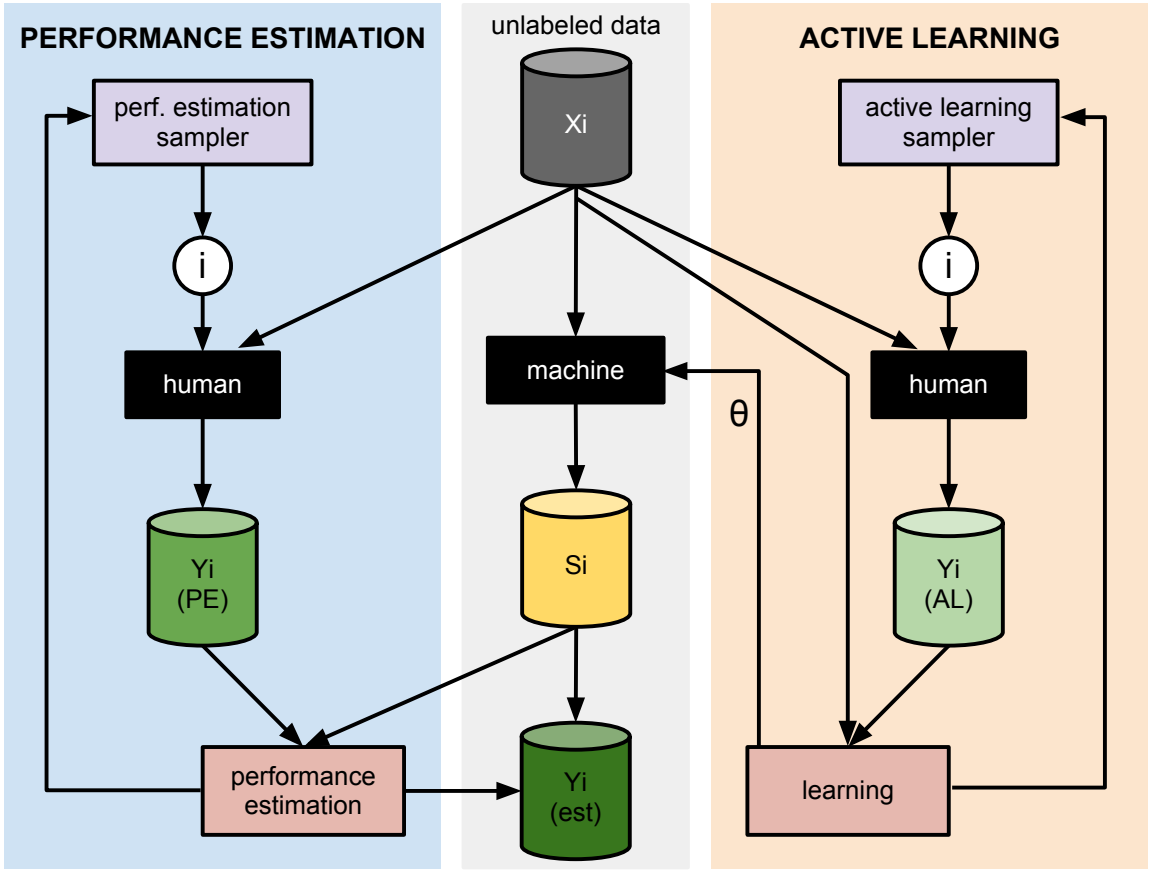

Figure 9.1: Combining classifier performance estimation with active learning. Unlabeled data X_i is annotated by humans (crowd or expert) to provide labels Y_i, and is used to train an active learning algorithm. The objective of the active learning algorithm is to choose training examples in an order that reduces the classifier error as quickly as possible. Simultaneously, some examples labeled by humans are used, together with the classifier scores S_i, to estimate the performance of the classifier. Once the classifier has reached the desired level of accuracy (with some confidence), the training can stop.

content that take up most of the space but which we do not understand, image and video data could become as accessible as textual data. The key to make it happen is to provide scalable image annotation systems, mostly made up of machine vision algorithms, but trained and quality controlled using crowds and experts. Hopefully, with the directions outlined in this thesis, that future is not far away.

Bibliography

[Att99] Hagai Attias. A Variational Baysian Framework for Graphical Models. In *NIPS*, pages 209–215, 1999.

[BBS10] T. Berg, A. Berg, and J. Shih. Automatic attribute discovery and characterization from noisy web data. *Computer Vision–ECCV 2010*, pages 663–676, 2010.

[BDL09] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *ICML*, 2009.

[Ben02] Paul N. Bennett. Using asymmetric distributions to improve classifier probabilities: A comparison of new and standard parametric methods. Technical report, Carnegie Mellon University, 2002.

[BG09] Gordon Bell and Jim Gemmell. *Total Recall: How the e-memory revolution will change everything*. Dutton, New York, 2009.

[BLNZ95] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[BM09] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.

[BRB+09] Kristin Branson, Alice A Robie, John Bender, Pietro Perona, and Michael H Dickinson. High-throughput ethomics in large groups of drosophila. *Nat Meth*, 6(6):451–457, 06 2009.

[BS87] Irving Biederman and Margaret M. Shiffrar. Sexing day-old chicks: A case study and expert systems analysis of a difficult perceptual-learning task. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(4):640–645, 1987.

[BS94] J. M. Bernardo and A. F. M. Smith. *Bayesian Theory.* Wiley, 1994.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Toward an Architecture for Never-Ending Language Learning. In *AAAI Conference on Artificial Intelligence*, 2010.

[CGJ96] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[Cox80] Eli P. Cox. The Optimal Number of Response Alternatives for a Scale: A Review. *Journal of Marketing Research*, 17(4):407–422, 1980.

[DDS+09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[DH08] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *ICML*, 2008.

[DLR+77] A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statistical Society, Series B*, 39(1):1–38, 1977.

[DS79] A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *J. Roy. Statistical Society, Series C*, 28(1):20–28, 1979.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *ICCV*, 2005.

[DWH+09] Heiko Dankert, Liming Wang, Eric D Hoopfer, David J Anderson, and Pietro Perona. Automated monitoring and analysis of social behavior in drosophila. *Nat Meth*, 6(4):297–303, 04 2009.

[DWP10] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *CVPR*, pages 1078–1085, 2010.

[DWSP11] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 99, 2011.

[ESCA06] Alaattin Erkanli, Minje Sung, E. Jane Costello, and Adrian Angold. Bayesian semi-parametric ROC analysis. *Statist. Med.*, 25:3905–3928, 2006.

[EVGW+10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[FA10] A. Frank and Arthur Asuncion. UCI machine learning repository, 2010.

[FFFP04] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *CVPR (Workshop on Generative-Model Based Vision)*, 2004.

[FFP05] Li Fei-Fei and Pietro Perona. A Bayesian hierarchical model for learning natural scene categories. In *CVPR*, pages 524–531. IEEE Computer Society, 2005.

[FHW+09] Thomas J. Fuchs, Johannes Haybaeck, Peter J. Wild, Mathias Heikenwalder, Holger Moch, Adriano Aguzzi, and Joachim M. Buhmann. Randomized tree ensembles for object detection in computational pathology. In George Bebis, Richard D. Boyle, Bahram Parvin, Darko Koracin, Yoshinori Kuno, Junxian Wang, Renato Pajarola, Peter Lindstrom, André Hinkenjann, Miguel L. Encarnacao, Claudio T. Silva, and

Daniel S. Coming, editors, *ISVC (1)*, volume 5875 of *Lecture Notes in Computer Science*, pages 367–378. Springer, 2009.

[FLW+08] Thomas J. Fuchs, Tilman Lange, Peter J. Wild, Holger Moch, and Joachim M. Buhmann. Weakly supervised cell nuclei detection and segmentation on tissue microarrays of renal cell carcinoma. In *Pattern Recognition. DAGM 2008*, volume 5096 of *Lecture Notes in Computer Science*, pages 173–182. Springer Berlin / Heidelberg, 2008.

[FPZ03] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, volume 2, pages 264–271, June 2003.

[GGR08] Jiezhun Gu, Subhashis Ghosal, and Anindya Roy. Bayesian bootstrap estimation of ROC curve. *Statist. Med.*, 27:5407–5420, 2008.

[GMT07] Gionis, Mannila, and Tsaparas. Clustering aggregation. In *ACM Transactions on Knowledge Discovery from Data*, volume 1. 2007.

[GS66] D.M. Green and J.M. Swets. *Signal detection theory and psychophysics*. John Wiley and Sons Inc, New York, 1966.

[GWKP11a] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *NIPS*, pages 558–566, 2011.

[GWKP11b] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. Technical Report CaltechAUTHORS:20110628-202526159, June 2011.

[HAS99] Martin Hellmich, Keith R. Abrams, and Alex J. Sutton. Bayesian Approaches to Meta-analysis of ROC Curves. *Med. Decis. Making*, 19:252–264, 1999.

[How06] Jeff Howe. The rise of crowdsoucing. *Wired*, 2006.

[JJ96] Tommi S. Jaakkola and Michael I. Jordan. A variational approach to Bayesian logistic regression models and their extensions, August 13 1996.

[Kru64] J. B. Kruskal. Multidimensional scaling by optimizing goodness-of-fit to a nonmetric hypothesis. *PSym*, 29:1–29, 1964.

[KWV07] Kenichi Kurihara, Max Welling, and Nikos Vlassis. Accelerated variational dirichlet process mixtures. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[LCGM09] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Turkit: tools for iterative tasks on mechanical turk. In *HCOMP*, pages 29–30, New York, NY, USA, 2009. ACM.

[LCGM10] Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Exploring Iterative and Parallel Human Computation Processes. In *HCOMP*, 2010.

[LDJ07] Tao Li, Chris H. Q. Ding, and Michael I. Jordan. Solving consensus and semi-supervised clustering problems using nonnegative matrix factorization. In *ICDM*, pages 577–582. IEEE Computer Society, 2007.

[Lik32] R Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140), 1932.

[Lin02] John M. Linacre. Optimizing Rating Scale Category Effectiveness. *Journal of Applied Measurement*, 3(1):85–106, 2002.

[LNH09] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer. In *CVPR*, 2009.

[Lo84] A.Y. Lo. On a class of bayesian nonparametric estimates: I. density estimates. *The Annals of Statistics*, pages 351–357, 1984.

[Lor80] F.M. Lord. *Applications of item response theory to practical testing problems*. Erlbaum, Mahwah, NJ, 1980.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[Mac92] David J.C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4:590–604, 1992.

[Mac94] Steven N. MacEachern. Estimating normal means with a conjugate style dirichlet process prior. *Communications in Statistics B*, 23(3):727–741, 1994.

[Mei03] M. Meila. Comparing clusterings by the variation of information. In *Learning theory and Kernel machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003: proceedings*, volume 2777, page 173. Springer Verlag, 2003.

[MMLM+09] G. Martinez-Munoz, N. Larios, E. Mortensen, W. Zhang, A. Yamamuro, R. Paasch, N. Payet, D. Lytle, L. Shapiro, S. Todorovic, et al. Dictionary-free categorization of very similar objects via stacked evidence trees. 2009.

[MTMG03] Stefano Monti, Pablo Tamayo, Jill Mesirov, and Todd Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1–2):91–118, 2003.

[Nea10] Radford M. Neal. MCMC using Hamiltonian dynamics. In S. Brooks, A. Gelman, G. L. Jones, , and X.-L. Meng, editors, *Handbook of Markov*

Chain Monte Carlo, pages 113–162. Chapman & Hall / CRC Press, 2010.

[NKP09] Vidhya Navalpakkam, Christof Koch, and Pietro Perona. Homo economicus in visual search. *J Vis*, 9(1):31.1–16, 2009.

[NMTM00] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[Par96] V. Pareto. Cours d'economie politique. 1896.

[Pla99] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74. MIT Press, 1999.

[Ras80] G Rasch. *Probabilistic models for some intelligence and attainment tests.* University of Chicago Press, Chicago, 1980.

[RTMF08] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vis.*, 77(1–3):157–173, 2008.

[RYZ+09] V.C. Raykar, S. Yu, L.H. Zhao, A. Jerebko, C. Florin, G.H. Valadez, L. Bogoni, and L. Moy. Supervised Learning from Multiple Experts: Whom to trust when everyone lies a bit. In *ICML*, 2009.

[See02] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2002.

[SF08] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *First IEEE Workshop on Internet Vision at CVPR'08*, 2008.

[SFB+95] P. Smyth, U. Fayyad, M. Burl, P. Perona, and P. Baldi. Inferring ground truth from subjective labelling of Venus images. *NIPS*, 1995.

[SG02] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[SOJN08] R. Snow, B. O'Connor, D. Jurafsky, and A.Y. Ng. Cheap and Fast - But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *EMNLP*, 2008.

[SP08] M. Spain and P. Perona. Some objects are more equal than others: measuring and predicting importance. In *ECCV*, 2008.

[SPI08] V.S. Sheng, F. Provost, and P.G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, 2008.

[SST09] I. Sutskever, R. Salakhutdinov, and J.B. Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[TK01] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, pages 45–66, 2001.

[TLB+11] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Kalai. Adaptively learning the crowd kernel. In *ICML*, pages 673–680, 2011.

[vAD04] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *SIGCHI conference on Human factors in computing systems*, pages 319–326, 2004.

[vAMM+08] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[VG09] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *CVPR*, pages 2262–2269, 2009.

[VG11] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds. In *CVPR*, 2011.

[VJ04] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[VRP10] Carl Vondrick, Deva Ramanan, and Donald Patterson. Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces. In *ECCV*, 2010.

[Wal45] A. Wald. Sequential tests of statistical hypotheses. *Ann. Math. Statist.*, 16(2):117–186, 1945.

[WBBP10] Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The Multidimensional Wisdom of Crowds. In *NIPS*, 2010.

[WBM+10] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[Wic02] T. D. Wickens. *Elementary signal detection theory*. Oxford University Press, United States, 2002.

[WP10] Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (ACVHL)*, 2010.

[WRW+09] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NIPS*, 2009.

[YRLT09] J. Yuen, B. C. Russell, C. Liu, and A. Torralba. LabelMe video: Building a Video Database with Human Annotations. In *ICCV*, 2009.

[Zhu08] Xiaojin Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin–Madison, 2008.