A Bayesian Concept Learning Approach to Crowdsourcing

Paolo Viappiani
Dept. of Computer Science
Aalborg University

Sandra Zilles, Howard J. Hamilton
Dept. of Computer Science
University of Regina

Craig Boutilier
Dept. of Computer Science
University of Toronto

Abstract

We develop a Bayesian approach to concept learning for crowdsourcing applications. A probabilistic belief over possible concept definitions is maintained and updated according to (noisy) observations from experts, whose behaviors are modeled using discrete types. We propose recommendation techniques, inference methods, and query selection strategies to assist a user charged with choosing a configuration that satisfies some (partially known) concept. Our model is able to simultaneously learn the concept definition and the types of the experts. We evaluate our model with simulations, showing that our Bayesian strategies are effective even in large concept spaces with many uninformative experts.

1 Introduction

Crowdsourcing is the act of outsourcing a problem to a group or a community. It is often referred to as *human computation*, as human "experts" are used to solve problems present difficulty for algorithmic methods; examples include Amazon's Mechanical Turk, the ESP game (for image labeling), and *reCaptcha* (for book digitization). Multiple human teachers, or experts, give feedback about (label) a particular problem instance. For instance, users refer to sites such as Yahoo! Answers to ask questions about everything from cooking recipes to bureaucratic instructions and health suggestions (e.g., which ingredients do I need to make tiramisu? how do I apply for a Chinese visa? how do I lose 20 pounds?).

As the information obtained with crowdsourcing is inherently noisy, effective strategies for aggregating multiple sources of information are critical. Aggregating noisy labels and controlling workflows are two problems in crowdsourcing that have recently been addressed with principled techniques [Dai *et al.*, 2010; Shahaf and Horvitz, 2010; Chen *et al.*, 2010]. In this work, we address the problem of generating recommendation for a user, where recommendation quality depends on some latent concept. The knowledge of the concept can only be refined by aggregating information from noisy information sources (e.g., human experts), and the user's objective is to maximize the quality of her choice as measured by satisfaction of the unknown latent concept. Achieving *complete* knowledge of the concept may be infeasible due to the quality of information provided by the experts, but also unnecessary. For instance, to successfully make tiramisu (a type of cake), certain ingredients might be necessary, while others may be optional. The concept c represents all possible "correct" recipes A configuration or instance x is a candidate recipe, and it satisfies c iff it can be used to make the cake (i.e., is correct). By asking various, possibly noisy, experts about particular ingredients, the user may "learn" a recipe satisfying c without ever learning *all* recipes satisfying c.

Following [Boutilier *et al.*, 2009], our aim is not to learn the concept definition *per se*; rather we want to learn just enough about it to make a (near-)optimal decision on the user's behalf. By exploiting the structure of the concept, a recommender system can adopt a strategy that queries only concept information that is relevant to the task at hand. For instance, if the system knows that an ingredient is extremely unlikely to be used in tiramisu, or is unlikely to be available, querying about this ingredient is unlikely to be helpful. Finally, the system needs to select the experts whose answers are (predicted to be) as informative as possible.

Our main contributions are 1) computational procedures to aggregate concept information (originating from noisy experts) into a probabilistic belief, 2) algorithms to generate recommendations that maximize the likelihood of concept satisfaction and 3) strategies to interactively select queries and experts to pose them to.

Our work is related to the model of Boutilier et al. [Boutilier *et al.*, 2009; 2010], who present a regret-based framework for learning subjective features in the context of preference elicitation. Our approach can be seen both as a Bayesian counterpart of that model, and as an extension to the case of multiple experts.

2 Bayesian Concept Learning Approach

We consider the problem of learning a latent concept by aggregating information from several sources called *experts*. Each expert may have a partial and incorrect definition of the concept. As in traditional concept learning [Mitchell0, 1977; Kearns and Li, 1993], we assume an abstract concept c is drawn from a concept class $\mathcal{C}$. However, instead of trying to identify the concept explicitly, we maintain a distribution

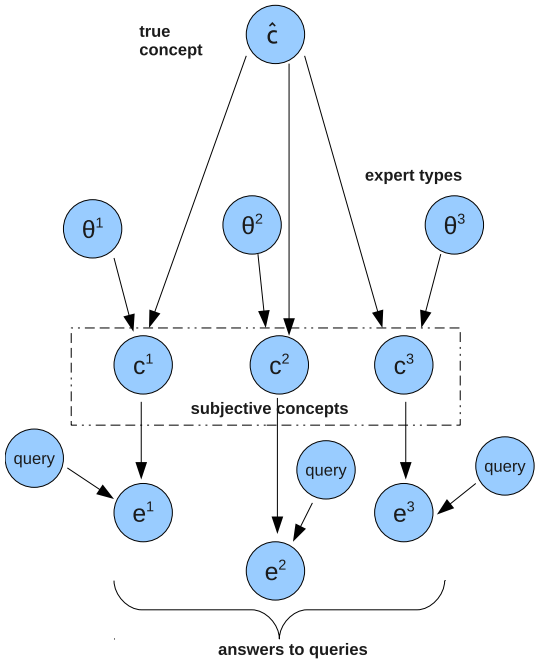

Figure 1: Abstract model of the problem of learning an unknown concept from mutiple noisy experts.

over possible concept definitions, and update the distribution according to the information acquired from the experts, in order to recommend an instance that is highly likely to satisfy the concept.

2.1 Concepts

We consider the problem of learning an abstract boolean concept drawn from a fixed concept class. A boolean concept c is a function $\{0,1\}^n \rightarrow \{0,1\}$, where $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ is a set of n boolean features. A *solution* (goal of the learning problem) is any boolean vector (configuration) $(x_1, \ldots, x_n) \in \{0,1\}^n$ for which $c(x_1, \ldots, x_n) = 1$. We allow the solution space to be restricted by feasibility constraints; below we assume linear constraints of the type $A \cdot x \leq B$ (with matrix A and vector B of the right dimensions). For example, *budget constraints* associate a vector of costs $(a_1, \ldots, a_n)$ with each feature and require total cost not to exceed the available budget b.

Throughout the paper, we restrict our focus to conjunctions [Haussler, 1989] as latent concepts, although our abstract model can be extended to boolean functions in general. A conjunctive concept c is a conjunction of literals over (some of) the atoms $\mathcal{X}_1, \ldots, \mathcal{X}_n$, e.g., $c = \mathcal{X}_2 \wedge \neg\mathcal{X}_4 \wedge \mathcal{X}_7$. A conjunction c can be equivalently represented as an assignment $(X_1^c, \ldots, X_n^c)$ of features $\mathcal{X}_1, \ldots, \mathcal{X}_n$ to the domain $\{T, F, DC\}$; in other words X_i^c can have one of the values T (true; the literal $\mathcal{X}_i$ occurs in c), F (false; the literal $\neg\mathcal{X}_i$ occurs in c), or DC (don't care; the atom $\mathcal{X}_i$ does not occur in c). In the above example, $X_2^c = X_7^c = T$, $X_4^c = F$, and $X_i^c = DC$ for $i \in \{2, 4, 7\}$.

Since the latter representation is used throughout the text, we write $c = (X_1^c, \ldots, X_n^c)$ and, with a slight abuse of notation, we will sometime refer to X_i^c as the "value" of feature i in concept c; we will also drop the superscript when c is clear from context. A configuration $x = (x_1, \ldots, x_n)$ yields $c(x) = 1$ (we say x satisfies c) iff (i) $x_i = 1$ for each i such that the literal $\mathcal{X}_i$ occurs in c ($X_i^c = T$), and (ii) $x_i = 0$ for each i such that the literal $\neg\mathcal{X}_i$ occurs in c ($X_i^c = F$).

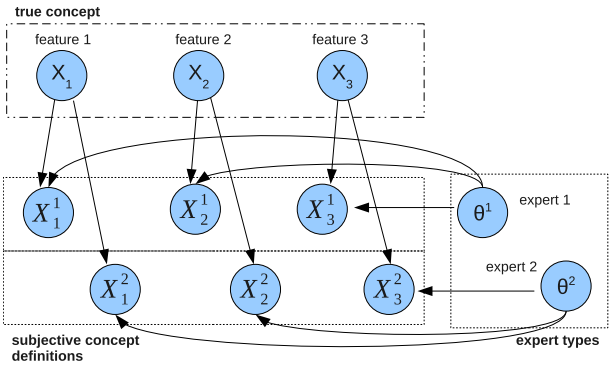

Figure 2: Graphical model for Bayesian learning of conjunctions (3 features, 2 experts.

Because the concept is unknown, the system maintains a belief $P(c) = P(X_1^c, \ldots, X_n^c)$. We assume some prior distribution over concepts. It is sometimes convenient to reason with the marginal probabilities, $P(X_i)$, representing the distribution over feature i, i.e., $P(X_i = T)$, $P(X_i = F)$, and $P(X_i = DC)$; for convenience, we write these terms as $P(T_i)$, $P(F_i)$, and $P(DC_i)$, respectively.

2.2 Query Types

The system acquires information about the concept by posing queries to a set of experts. These concept queries can be of different forms (e.g., membership, equivalence, superset, or subset queries [Angluin, 1988]) and their answers partition the hypothesis space. For instance, a membership query asks whether a given configuration x satisfies the concept (e.g., *"Is this a valid recipe for tiramisu?"*). Membership queries can be too cognitively demanding for a crowd-sourcing domain, as an expert would have to verify every problem feature to check whether the provided instance is satisfied. Thus, in this work we focus on *literal* queries, a special form of *superset queries*. A literal query q_i on feature i asks for the value of X_i; possible answers to the query are T, F, or DC.[1] Literal queries can be thought of as requests for a piece of information such as *"Are eggs needed for tiramisu?"*. Query strategies for selecting literal queries are discussed in Section 4.[2]

2.3 Expert Types

In practice, experts do not always provide correct answers. Hence we assume that experts belong to different populations

[1] Alternatively, one could ask queries such as "Is X_i positive in the concept definition?" Adapting our model to such queries is straightforward.

[2] Notice that literal queries cannot be answered unambiguously in general since dependencies may exist; but the value of a literal in a conjuctive concept is independent of the value of any other literal.

or *types* from a set $\mathcal{T} = \{t_1, \ldots, t_k\}$. The type of an expert represents the expert's capacity and commitment to correctly answering queries about the concept (or aspects thereof). For instance, as in [Chen *et al.*, 2010], types might discriminate "good" or knowledgeable experts, whose answers are likely to be correct, from "bad" experts, whose answers are drawn randomly. Our model generalizes to any number of types.

We indicate the assignments of types to experts with a vector $\theta = (\theta^1, \ldots, \theta^m)$, where $\theta^j \in \mathcal{T}$ is the type of expert j. A further natural assumption is that experts are noisy and provide feedback with respect to their subjective definition of the concept. In other words, we assume that there exists one underlying (true) concept definition $\hat{c} = (X_1, \ldots, X_n)$, but each expert's response is based on its own subjective concept $c^j = (X_1^j, \ldots, X_n^j)$. When a query q_i^j on feature i is posed to expert j, the expert reveals its subjective value X_i^j for that feature (either T, F or DC). Subjective concepts are distributed, in turn, according to a generative model $P(c^j|\hat{c}, \theta^j)$, given expert type θ^j and true concept $\hat{c}$. For example, an "uninformed" expert may have a subjective concept that is probabilistically independent of $\hat{c}$, while an "informed" expert may have a concept that is much more closely aligned with $\hat{c}$ with high probability. In our experiments below, we assume a factored model $P(X_i^j|X_i, \theta^j)$. Moreover, since we always ask about a specific literal, we call this distribution the *response model*, as it specifies the probability of expert responses as function of their type. This supports Bayesian inference about the concept given expert answers to queries (note that we do not assume expert types are themselves observed; inference is also used to estimate a distribution over types).

The graphical model for the general case is shown in Figure 1. In Figure 2 we show the model for conjunctions with 3 features and 2 experts; the subjective concept c^j of expert $j \in \{1, 2\}$ is composed by X_1^j, X_2^j and X_3^j.

As queries provide only "noisy" information about the true concept $\hat{c}$, the system cannot fully *eliminate* hypotheses from the version space given expert responses. To handle concept uncertainty, the system maintains a distribution or *belief* $P(c)$ over concept definitions, as well as a distribution over expert types $P(\theta)$. Both distributions are updated whenever queries are answered.

Beliefs about the true concept and expert subjective concepts will generally be correlated, as will beliefs about the types of different experts. Intuitively, if two experts consistently give similar answers, we expect them to be of the same type. When we acquire additional evidence about the type of one expert, this evidence affects our belief about the type of the other expert as well. Thus, when new evidence e is acquired, the joint posterior $P(c, \theta|e)$ cannot be decomposed into independent marginals over c and the θ^j, since c and θ are not generally independent. Similarly, new evidence about feature X_i might change one's beliefs about types, and therefore influence beliefs about another feature X_j. We discuss the impact of such dependence on inference below.

2.4 Decision-making

The system needs to recommend a configuration $x = (x_1, \ldots, x_n) \subseteq \{0, 1\}^n$ that is likely to satisfy the concept (e.g., a recipe for tiramisu), based on the current belief $P(c)$. A natural approach is to choose a configuration x^* that maximizes the *a posteriori* probability of concept satisfaction (MAPSAT) according to the current belief: $x^* \in \arg\max_x P(c(x))$.

Exact maximization typically requires enumerating all possible configurations and concept definitions. Since this is not feasible, we consider the marginalized belief over concept features and optimize, as a surrogate, the product of probabilities of the individual features satisfying the configuration: $P(c(x)) \approx \tilde{P}(c(x)) = \prod_i P(c_i(x_i))$, where c_i is the restriction of concept c to feature i. In this way, optimization without feasibility or budget constraints can be easily handled. For each feature i, we choose $x_i = 1$ whenever $P(T_i) \geq P(F_i)$, and choose $x_i = 0$ otherwise.

However, in the presence of feasibility constraints, we cannot freely choose to set attributes in order to maximize the probability of concept satisfaction. We show how, using a simple reformulation, this can be solved as an integer program. Let $p_i^+ = P(T_i) + P(DC_i)$ be the probability that setting $x_i = 1$ is consistent with the concept definition for the i-th feature; similarly let $p_i^- = P(F_i) + P(DC_i)$ be the probability that setting $x_i = 0$ is consistent. Then the probability of satisfying the i-th feature is $P(c_i(x_i)) = p_i^+ x_i + p_i^-(1 - x_i)$. The overall (approximated) probability of concept satisfaction can be written as:

$$P(c(x)) \approx \prod_{1 \leq i \leq n} p_i^+ x_i + p_i^-(1 - x_i) = \prod_{1 \leq i \leq n} (p_i^+)^{x_i} \prod_{1 \leq i \leq n} (p_i^-)^{(1-x_i)} \quad (1)$$

The latter form is convenient because we can linearize the expression by applying logarithms. To obtain the feasible configuration x^* maximizing the probability of satisfaction, we solve the following integer program (the known term has been simplified):

$$\max_{x_1, \ldots, x_n} \sum_{1 \leq i \leq n} [log(p_i^+) - log(p_i^-)] \cdot x_i \quad (2)$$

$$s.t. \quad A \cdot x \leq B \quad (3)$$

$$x \in \{0, 1\}^n \quad (4)$$

3 Inference

When a query is answered by some expert, the system needs to update its beliefs.Let e_i^j represent the evidence (query response) that expert j offers about feature i. Using Bayes' rule, we update the probability of the concept: $P(c|e_i^j) \propto P(e_i^j|c)P(c)$. Since the type θ^j of expert j is also uncertain, inference requires particular care. We consider below several strategies for inference. When discussing their complexity, we let n denote the number of features, m the number of experts, and k the number of types.

Exact Inference Exact inference is intractable for all but the simplest concepts. A naive implementation of exact inference would be exponential in both the number of features

and the number of experts. However, inference can be made more efficient by exploiting the independence in the graphical model. Expert types are mutually independent given concept c: $P(\theta|c) = \prod_{1 \leq j \leq m} P(\theta^j|c)$. This means that each concept can be "safely" associated with a vector of m probabilities $P(\theta^1|c), \ldots, P(\theta^m|c)$, one for each expert. For a concept space defined over n features, we explicitly represent the 3^n possible concept definitions, each associated with a matrix (of dimension m by k) representing $P(\theta|c)$. The probability of a concept is updated by multiplying the likelihood of the evidence and renormalizing: $P(c|e_i^j) \propto P(e_i^j|c)P(c)$. As the queries we consider are *local* (i.e., only refer to a single feature), the likelihood of c is

$$P(e_i^j|c) = \sum_{t \in \mathcal{T}} P(e_i^j|\theta^j = t, X_i^c) P(\theta^j = t|c), \tag{5}$$

where X_i^c is the value of c for feature $\mathcal{X}_i$. The vector $(P(\theta^1|c, e_i^j), \ldots, P(\theta^m|c, e_i^j))$ is updated similarly. Overall complexity is $O(3^n mk)$. Since the number of experts m is usually much larger than the number of features n, exact inference is possible for small concept spaces. In practice, it is only feasible for up to 5–10 features; in our implementation, exact inference with $n = 7$ and $m = 100$ requires 3–4 seconds per query.

Naive Bayes This approach to inference makes the strong assumption that X_i and θ^j are mutually conditionally independent. This allows us to factor the concept distribution into marginals over features: $P(X_1), \ldots, P(X_n)$; similarly beliefs about experts are represented as $P(\theta^1), \ldots, P(\theta^m)$. The likelihood $P(e_i^j|X_i)$ of an answer to a query can be related to $P(e_i^j|\theta^j, X_i)$ (the response model) by marginalization over the possible types of expert j: $P(e_i^j|X_i) = \sum_{v \in \{t_1, t_2, \ldots\}} P(e_i^j|\theta^j = v, X_i) P(\theta^j = v|X_i)$. We write the expression for the updated belief about X_i given evidence:[3]

$$P(X_i|e_i^j) = \frac{P(e_i^j|X_i)P(X_i)}{P(e_i^j)} \tag{6}$$

$$= \frac{\sum_{t \in \mathcal{T}} P(e_i^j|X_i, \theta^j{=}t) P(\theta^j, X_i)}{\sum_{z \in \{T,F,DC\}} \sum_{t \in \mathcal{T}} P(e_i^j|X_i{=}z, \theta^j{=}t) P(\theta^j, X_i)} \tag{7}$$

We update belief $P(X_i)$ using current type beliefs $P(\theta^1), \ldots, P(\theta^m)$. Our strong independence assumption allows simplification of Eq. 7:

$$P(X_i|e_i^j) = \frac{\sum_{t \in \mathcal{T}} P(e_i^j|X_i, \theta^j{=}t) P(\theta^j{=}t)}{\sum_z \sum_{t'} P(e_i^j|X_i{=}z, \theta^j = t') P(\theta^j{=}t') P(X_i{=}z)} P(X_i) \tag{8}$$

Similarly, for beliefs about types we have:

$$P(\theta^j|e_i^j) = \frac{\sum_z P(e_i^j|X_k = z, \theta^j) P(X_i = z)}{\sum_{z'} \sum_t P(e_i^j|X_i = z', \theta^j = t) P(\theta^j) P(X_i = z')} P(\theta^j) \tag{9}$$

This approximation is crude, but performs well in some settings. Moreover, with space complexity $O(n + m)$ and time complexity $O(nm)$, it is very efficient.

[3]Using Naive Bayes, we only update concept beliefs about X_i, the feature we asked about. Similarly, for types, we only update relative to θ^j, the expert that answered the query.

Monte Carlo This approximate inference technique maintains a set of l particles, each representing a specific concept definition, using importance sampling. As with exact inference, we can factor beliefs about types. The marginal probability $P(X_i)$ that a given feature is true in the concept is approximated by the fraction of the particles in which X_i is true (marginalization over types is analogous). Whenever queries are answered, the set of particles is updated recursively with a resampling scheme. Each particle is weighted by the likelihood of the concept definition associated with the particle when evidence e_k^u is observed (the higher the likelihood, the higher the chance of resampling). Formally, the expression of the likelihood of a particle is analogous to the case of exact inference, but we only consider a limited number of possible concepts. Monte Carlo has $O(lmk)$ complexity; hence, it is more expensive than Naive Bayes but less expensive than exact inference.

4 Query Strategies

We now present elicitation strategies for selecting queries. Each strategy is a combination of methods that, given the current beliefs about the concept and the types, i) selects a feature to ask about, and ii) selects the expert to ask. Expert selection depends on the semantics of the types; here, as in [Chen *et al.*, 2010], we assume experts are either "knowledgeable" (type t_1) or "ignorant" (type t_2). As baseline, we consider two inefficient strategies for comparison purposes: (i) *broadcast* iterates over the features and, for each, asks the same query to a fixed number of experts, and (ii) *dummy* asks random queries of random experts and recommends the most frequent answers.

Feature Selection We consider three strategies aimed at directly reducing concept uncertainty. The *maximum entropy* (or *maxent*) strategy selects the feature whose probability distribution over $\{T, F, DC\}$ has the greatest entropy. Unfortunately, this measure treats being uncertain between a T and F as the same as being uncertain between T and DC. The *minval* strategy selects the feature $\mathcal{X}_f$ with the lowest probability of "getting it right:" that is, $f = \arg\min_i \{\max(p_i^+, p_i^-)\}$ is viewed as the feature with the greatest potential for improvement. Each feature is "scored" using the probability, given our current beliefs, that the best guess for its feature value will match the true concept. The intention is to reduce the uncertainty that most hinders the chance of satisfying the concept. Finally, queries can be evaluated with respect to their capacity to improve decision quality using value of information [Howard, 1966]. We optimize *expected value of perfect information (EVPI)*; as shown below, this criterion can be computed using the current belief without expensive Bayesian updates. In this setting, *EVPI* measures the expected gain in the quality of a decision should we have access to perfect information about a particular feature. In other words, given an oracle able to provide the actual value (T, F or DC) of a feature, which should we ask about? The value

of querying feature X_i is:[4]

$$EVPI_i = \sum_{z \in \{T,F,DC\}} P(X_i = z) \max_x P(c(x)|X_i = z). \quad (10)$$

Since we aim to select queries quickly, we also consider *Naive EVPI*, where $P(c(x)|X_i)$ is approximated by the product of satisfying each feature.

Observation 1 *In unconstrained problems, the feature selected with the minval heuristic strategy is associated with maximum Naive EVPI.*

The proof is provided in the Appendix. It relies on the fact that, without feasibility constraints, one can optimize features independently. For the more general case, given feature i, we define $x^{+i} = \arg\max_{x \in X: x_i = 1} P(c(x))$ to be the optimal configuration among those where feature i is true; we define x^{-i} analogously. We write the approximated satisfaction probabilities as $\tilde{P}(c(x^{+i})) = p_i^+ \cdot p_{\neq i}^{+i}$, where $p_{\neq i}^{+i} = \prod_{j \neq i} P(c_j(x^{+i}))$, and $\tilde{P}(c(x^{-i})) = p_i^- \cdot p_{\neq i}^{-i}$.

Observation 2 *Naive* EVPI *can readily be computed using the current belief:*

$$EVPI_i = P(T_i)p_{\neq i}^{+i} + P(F_i)p_{\neq i}^{-i} + P(DC_i) \cdot \max\{p_{\neq i}^{+i}, p_{\neq i}^{-i}\}$$

From this Observation it follows that, if $P(DC_i) = 0$ (we know that a feature is either true or false in the concept definition), then $EVPI_i = \tilde{P}(c(x^{+i})) + \tilde{P}(c(x^{-i}))$. The most informative feature is the feature i that maximizes the sum of the probability of concept satisfaction of x^{+i} and x^{-i}. This, in particular, is true when one considers a concept space where "don't care" is not allowed.

Naive *EVPI* query maximization is in general very efficient. As the current best configuration x^* will coincide with either x^{i+} or x^{i-} for any feature i, it requires only $n + 1$ MAPSAT-optimizations and n evaluations of $EVPI$ using Observation 2. Its computational complexity is not affected by the number of experts m.

Expert Selection For a given feature, the *greedy* strategy selects the expert with the highest probability of giving an informative answer (i.e., one of type t_1). It is restricted to never ask the the same expert about the same feature, which would be useless in our model. However, there can be value in posing a query to an expert other than that predicted to be most "knowledgeable" because we may learn more about the types of other experts. The *soft-max* heuristic accomplishes this by selecting an expert j according to a Boltzmann distribution $\frac{e^{P(\theta^j = t_1)/\tau}}{\sum_r e^{P(\theta^r = t_1)/\tau}}$ with "temperature" τ, so that experts that are more likely to be of type t_1 are queried more often.

[4]We consider each possible response (T, F or DC) by the oracle, the recommended configuration conditioned to the oracle's answer, and weight the results using the probability of the oracle's response.

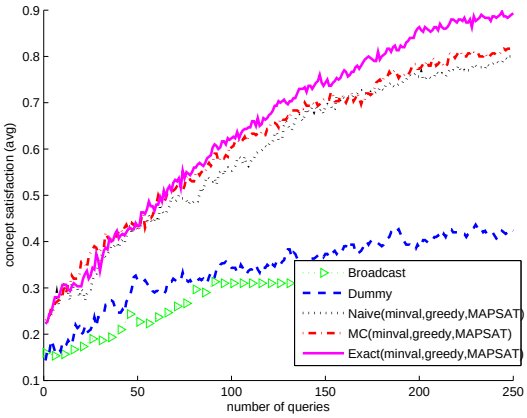

Figure 3: Simulation with 5 features, 100 experts (20% knowledgeable experts); 300 runs

Combined Selection There can be value in choosing the feature and expert to ask in combination. We consider strategies inspired by work on multi-armed bandit problems [Sutton and Barto, 1998], aimed at resolving the tradeoff between *exploration* and *exploitation*. In this setting, exploitation means using a strategy such as *EVPI* to learn more about the concept; in this case, we select experts greedily. On the other hand, exploration in this context means using a strategy such as *soft-max* to *learn more* about expert types; in this case, we select the feature we are most certain about because it will provide the most information about an expert's type. The *explore-exploit* strategy embodies this tradeoff: we generate the pair (i, j), where $\mathcal{X}_i$ is the feature that maximizes *EVPI* and j is the expert chosen greedily as above. We then consider our current belief $P(\theta^j)$ about its type and use this to switch between exploitation and exploration. We sample a value from $P(\theta^j)$; if we obtain t_1, we query q_i^j (exploitation), otherwise, we generate (i', j'), where i' is the index of the feature we are most certain about and j' is chosen with *soft-max* (exploration). In practice this method is more effective using a Boltzmann distribution over types; in the experiments below we "exploit" with probability $0.5 + 0.5 * \frac{e^{P(\theta^j = t_1)/\tau}}{e^{P(\theta^j = t_1)/\tau} + e^{P(\theta^j = t_2)/\tau}}$.

5 Experiments

We experimented with the query strategies described in Section 4 by comparing their effectiveness on randomly generated configuration problems and concepts. Queries are asked of simulated experts, each with a type and a subjective concept drawn from a prior distribution.[5] At any stage, each strategy recommends a configuration (decision) based on the

[5]The type is either "knowledgeable" or "ignorant." We define probabilities for subjective concept definitions such that 70% of the time, knowledgeable experts reveal the true value of a particular feature (i.i.d. over different features), and a true T value is reported to be DC with higher probability than is F. Ignorant experts are uniformative (in expectation) with each feature value T, F, and DC sampled

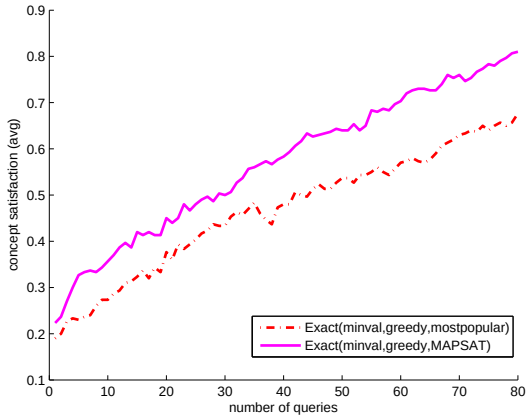

Figure 4: MAPSAT vs *mostpopular* (5 features, 100 experts, 30% knowledgeable, 300 runs)

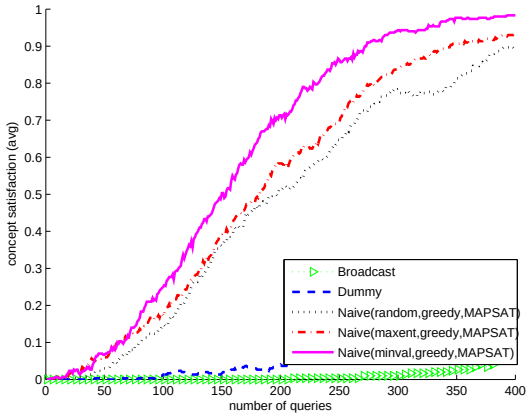

Figure 5: Evaluation of feature selection methods in a larger concept space (30 features; 50% knowledgeable; 500 runs)

current belief and selects the next query to ask; we record whether the current configuration satisfies the true concept.

The concept prior (which is available to the recommender system) is sampled using independent Dirichlet priors for each feature; this represents cases where prior knowledge is available about which features are most likely to be involved (either positively or negatively) in the concept. A *strategy* is a combination of: an inference method; a heuristic for selecting queries (feature and expert); and a method for making recommendations (either MAPSAT or *mostpopular*, the latter a heuristic that recommends each configuration feature based on the most common response from the experts).

Our results below show that good recommendations can be offered with very limited concept information. Furthermore, our decision-theoretic heuristics generate queries that allow a

from a random multinomial, drawn from a Dirichlet prior *Dir(4,4,4)*. Since an expert's answers are consistent with its subjective concept, repeating a query to some expert has no value.

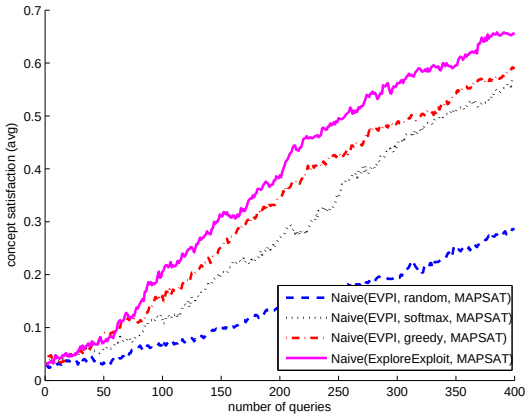

Figure 6: Evaluation of expert selection methods (20 features; 20% of experts are knowledgeable; 500 runs)

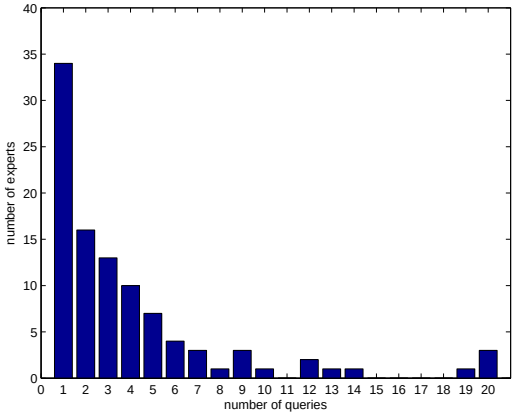

Figure 7: Distribution of the number of queries posed to experts

concept-satisfying recommendation to be found quickly (i.e., with relatively few expert queries). In the first experiment (see Figure 3), we consider a setting with 5 features and 100 experts, and compare all methods for Bayesian inference (Exact, Naive and Monte Carlo with 100 particles). All three methods generate queries using *minval* (to select features) and *greedy* (to select experts). We also include *broadcast* and *dummy*. Only 20% of experts are knowledgeable, which makes the setting very challenging, but potentially realistic in certain crowdsourcing domains. Nonetheless our Bayesian methods identify a satisfactory configuration relatively quickly. While the exact method performs best, naive inference is roughly as effective as the more computationally demanding Monte Carlo strategy, and both provide good approximations to Exact in terms of recommendation quality. *Dummy* and *broadcast* perform poorly; one cannot expect to make good recommendations by using a simple "majority rule" based on answers to poorly selected queries. In a similar setting, we show that MAPSAT outperforms *mostpopular* (assign features based on the most frequent answers) for

choosing the current recommendation also when used with exact inference (Figure 4).[6]

In the next experiment, we consider a much larger concept space with 30 boolean variables (Figure 5). In this more challenging setting, exact inference is intractable; so we use Naive Bayes for inference and compare heuristics for selecting features for queries. *Minval* is most effective, though maxent and random perform reasonably well.

Finally we evaluate heuristics for selecting experts (*random*, *greedy* and *softmax*) and the combined strategy (*explore-exploit*) in presence of budgeted constraints. Each feature is associated with a cost a_i uniformly distributed between 1 and 10; this cost is only incurred when setting a feature as positive (e.g. when buying an ingredient); the available budget b is set to $0.8 \cdot \sum_i a_i$.

Figure 6 shows that the *explore-exploit* is effective and outperforms the other strategies. This suggests that our combined method balances exploration (asking queries in order to know more about the type of the experts) and exploitation (asking the query to the must knowledgeable expert given our belief) in an effective way. It is interesting to observe that *Naive(EVPI,greedy,MAPSAT)*, while using the same underlying heuristic for selecting features as *Naive(explore-exploit,MAPSAT)*, is very effective at the beginning but becomes outperformed after approximately 50-60 queries, as it never explicitly tries to ask queries aimed at improving knowledge about the expert types.

Although the number of queries may seem large, they are asked of different experts; a single expert is asked at most n queries, and most experts are asked only 1 or 2 queries. Figure 7 shows a histogram about the number of queries asked to experts by *explore-exploit* in the last setting: 3 experts are asked 20 queries, while 34 experts are asked only one.

6 Discussion and Future Work

We have presented a probabilistic framework for learning concepts from noisy experts in a crowdsourcing setting, with an emphasis on learning just enough about the concept to identify a concept instance with high probability. We described methods for making recommendations given uncertain concept information and how to determine the most "relevant" queries. Since experts are noisy, our methods acquire indirect information about their reliability by aggregating their responses to form a a distribution over expert types. Our experiments showed the effectiveness of our query strategies and our methods for approximate inference, even in large concept spaces, with many uninformative experts, and even when "good" experts are noisy.

The are many interesting future directions. Development of practical applications and validation with user studies is of critical importance. While we have focused on conjunctive concepts in this paper, we believe our model can be extended to more general concept classes. Special care, however, must be taken in several aspects of an extended model: the exact semantics of queries; the representation of the concept distribution; and inference over types and concepts. We are also interested in a game-theoretic extension of the model that allow (some or all) experts to provide responses that reflect their self-interest (e.g., by guiding a recommender system to specific products).

Further investigation of query selection strategies is important; our strategies adopt ideas from multi-armed bandit and we are interested in exploring this connection in more details. Principled methods for query optimization in preference elicitation [Viappiani and Boutilier, 2010] could also provide additional insights.

Our model values configurations based on their probability of satisfying the concept (i.e., assuming binary utility for concept satisfaction). Several other utility models can be considered. For instance, we might define utility as a sum of some concept-independent reward for a configuration—reflecting user preferences over features that are independent of the latent concept—plus an additional reward for concept satisfaction (as in [Boutilier *et al.*, 2009; 2010]). One could also consider cases in which it is not known with certainty which features are available: the problem of generating recommendations under both concept and availability uncertainty would be of tremendous interest.

[6] As our heuristics only ask queries that are relevant, recommendations made by the *mostpopular* strategy are relatively good in this case.

References

[Angluin, 1988] D. Angluin. Queries and concept learning. *Mach. Learn.*, 2:319–342, 1988.

[Boutilier *et al.*, 2009] C. Boutilier, K. Regan, and P. Viappiani. Online feature elicitation in interactive optimization. In *ICML 2009*, pages 73–81, 2009.

[Boutilier *et al.*, 2010] C. Boutilier, K. Regan, and P. Viappiani. Simultaneous elicitation of preference features and utility. In *AAAI 2010*, pages 1160–1197, 2010.

[Chen *et al.*, 2010] S. Chen, J. Zhang, G. Chen, and C. Zhang. What if the irresponsible teachers are dominating? In *AAAI 2010*, 2010.

[Dai *et al.*, 2010] P. Dai, Mausam, and D.S. Weld. Decision-theoretic control of crowd-sourced workflows. In *AAAI 2010*, 2010.

[Haussler, 1989] D. Haussler. Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4:7–40, 1989.

[Howard, 1966] Ronald Howard. Information value theory. *IEEE Trans. on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[Kearns and Li, 1993] M.J. Kearns and M. Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22:807–837, 1993.

[Mitchell0, 1977] T.M. Mitchell0. Version spaces: A candidate elimination approach to rule learning. In *IJCAI 1977*, pages 305–310, 1977.

[Shahaf and Horvitz, 2010] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *AAAI 2010*, 2010.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, 1998.

[Viappiani and Boutilier, 2010] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, pages 2352–2360. MIT press, 2010.

Proof of Observation 1: Assume we ask the oracle about feature i. Let $p_i^* = \max(p_i^+, p_i^-)$. The optimal configuration x^* *in the updated belief* given the oracle's response is such that $x^* = \arg\max_x \prod_i P(c_i(x)|X_i{=}v)$, where v (either T,F or DC) is the oracle's response. Since there are no constraints, it can be optimized independently for the different features. Feature i of the optimal configuration x_i^* will necessarily be set to 1 or 0 in a way consistent with v (in case of DC, either is equivalent) and we are sure that x_i^* satisfies feature i; all other features will be set according to p_i^*. The (approximated) probability of concept satisfaction is:

$$\max_x \prod_j P(c_j(x)|X_i{=}v) = \prod_{j\neq i} \max(p_j^+, p_j^-) = \prod_{j\neq i} p_i^* = p_{\neq i}^*. \tag{11}$$

Therefore, $EVPI_i = \sum_{v=T,F,DC} P(X_i = v) \cdot p_{\neq i}^* = p_{\neq i}^*$. The argument follows from observing that $i = \arg\max p_{\neq i}^*$ iff $i = \arg\min p_i^*$. ∎

Proof of Observation 2: Note that x^{+i} and x^{-i} are the optimal configurations in the posterior beliefs $P(c|X_i = T)$ and $P(c|X_i = F)$ respectively. In the case that the oracle's answer is DC ("don't care") then the optimal configuration is either x^{+i} or x^{-i} depending on which of the two gives higher probability of satisfying all features beside i. The argument follows from Equation 10. ∎