Optimal and Efficient Binary Questioning for Accelerated Annotation

Franco Marchesoni-Acland[1,3], Jean-Michel Morel[2], Josselin Kherroubi[1], Gabriele Facciolo[3]

[1]SLB, AI Lab, Paris
[2]City University of Hong Kong
[3]Université Paris-Saclay, ENS Paris-Saclay, CNRS, Centre Borelli, France
marchesoniacland@gmail.com

Abstract

Even though data annotation is extremely important for interpretability, research, and development of artificial intelligence solutions, annotating data remains costly. Research efforts such as active learning or few-shot learning alleviate the cost by increasing sample efficiency, yet the problem of annotating data more quickly has received comparatively little attention. Leveraging a predictor has been shown to reduce annotation cost in practice but has not been theoretically considered. We ask the following question: to annotate a binary classification dataset with N samples, can the annotator answer less than N yes/no questions? Framing this question-and-answer (Q&A) game as an optimal encoding problem, we find a positive answer given by the Huffman encoding of the possible labelings. Unfortunately, the algorithm is computationally intractable even for small dataset sizes. As a practical method, we propose to minimize a cost function a few steps ahead, similarly to lookahead minimization in optimal control. This solution is analyzed, compared with the optimal one, and evaluated using several synthetic and real-world datasets. The method allows a significant improvement ($23-86\%$) in the annotation efficiency of real-world datasets.

Introduction

Deep supervised learning has been vital in advancing artificial intelligence (AI) and solving practical problems. Even now, with more powerful self-supervised methods, human feedback is essential to ensure proper deployment and correct downstream performance (Bai et al. 2022; Oquab et al. 2023). Manual supervision allows engineers to fine-tune powerful models for downstream tasks. Even though datasets are fundamental for AI research and development, their quick annotation is still an open problem.

Much research focused on obtaining better predictors given some annotated data (e.g. ImageNet (Beyer et al. 2020)). More recently, sampling efficiency considerations have gained attention, as seen in active learning (AL), few-shot learning (FSL), or semi-supervised learning. Simultaneously, data availability has been considered in self-supervised or weakly-supervised learning fields. All these areas study the path from data to predictors. In contrast, comparatively fewer works study the path from predictors to data, even though it has been used in practice (see below). Table 1 shows a simplified comparison between different learning-related tasks. Tools that reduce the annotation cost of one sample can speed up the whole process, as illustrated by interactive image/video segmentation (Liu et al. 2024; Ravi et al. 2024). But what if one bit is enough to annotate one sample? Can we speed up annotation even in that case? The present work theoretically shows that the answer is positive when a predictor is available. Although the theoretically optimal solution is computationally intractable, our proposed approximation is efficient and performs well. Ultimately, we provide an algorithm that asks the best binary questions to the annotator. As exposed in Section , the method is based on asking questions involving sets of samples, for example, *"are these m samples positive?"*. The effectiveness of the method relies on the annotator's ability to assess multiple samples simultaneously; therefore, it may be less useful for tasks such as text classification, and more useful for tasks such as audio classification. Rooted in theory, our proposed method helps to better understand ad-hoc methods and think about their optimality. As shown in Section , the algorithm significantly improves the annotation efficiency in the oracle setting by up to 88% and in the most realistic scenario by $\approx 70\%$.

Related Work

Theory on Binary Questioning: There are two notable theoretical analyses of this problem. The first (Kulkarni, Mitter, and Tsitsiklis 1993) studies the learnability bounds of an active learner that asks yes/no questions. This problem is related to source coding in information theory (see below). However, their learner wants to minimize the maximum number of questions (length of the longest codeword), whereas we care about the expected number of questions (mean codeword length). The second relevant theoretical work (Yang, Hanneke, and Carbonell 2010) also deals with (active) learning bounds. Again, the authors point out the equivalence between the problem and coding theory but choose to consider lossy coding algorithms. The biggest theoretical difference between our work and these papers (Kulkarni, Mitter, and Tsitsiklis 1993; Yang, Hanneke, and Carbonell 2010) is that they focus on lossy compression for quick learning, while we focus on lossless compression for quick annotation. The notation we later introduce is simpler

	Few-shot	Active	Human-in-the-loop	Ours
Maximizes	Performance	Performance	Performance	Number of annotated samples
vs.	Number of (given) annotated samples	Number of (chosen) annotated samples	Number of user answers	Number of user answers

Table 1: Characterization of different types of learning tasks depending on their objective.

than the one in (Kulkarni, Mitter, and Tsitsiklis 1993), which leverages the notion of *concept classes*.[1]

Faster Annotation Methods: The SAM series (Kirillov et al. 2023; Ravi et al. 2024) exemplifies the power of faster annotation, publishing two datasets that are the largest in their respective areas. They introduce a data engine and an interactive segmentation (IS) model that lowers the annotation time for each mask and also for each image/video. They simultaneously reduce the time to annotate each instance and leverage pseudo-labeling equipped with manual checks and corrections. They work on the specific area of image/video segmentation. More related to our work are ad-hoc methods such as (Xie et al. 2022; Hu et al. 2020) that introduce yes/no answers as supervision. Notably, (Hu et al. 2020) uses binary questioning to reduce the annotation cost for multiclass classification. The authors of this paper make the annotator answer a binary question by checking the correctness of a guess instead of requiring $\log_2 C$ bits to select one of C classes. Moreover, they propose a method to incorporate the information of an incorrect guess when it happens. Similarly, (Xie et al. 2022) aims to reduce the number of questions for annotating an observation in image segmentation. They propose a method to choose pixels to be queried based on *region impurity and prediction uncertainty* that performs better than all other competitors, the second best being random sampling. All these works reduce the annotation cost of one instance. This is impossible in the binary classification setting, as the annotation cost for a single observation is always 1 bit. The only possible way to reduce the annotation cost in the binary classification setting is by considering multiple observations simultaneously. This concept is orthogonal to the problem of reducing the annotation cost of one instance.

Dynamic Programming (DP): DP is an optimization algorithm widely used to solve optimal control/reinforcement learning problems. The closest to our work is the recent Wordle resolution (Bhambri, Bhattacharjee, and Bertsekas 2022) based on the lookahead minimization method whose formalization was popularized by Bertsekas (Bertsekas 2012, 2022). One-step lookahead minimization is guaranteed to improve upon an initial policy (Bertsekas 2021). This process, equipped with learned cost functions, is core to the well-known AlphaZero algorithm (Silver et al. 2018). In contrast to AlphaZero, our algorithm can not learn cost functions as it only has one question-answering "game" to play. Luckily, heuristics arise naturally and facilitate an initial policy both in Wordle (Bhambri, Bhattacharjee, and Bertsekas 2022) and in our problem.

Active Learning (AL): AL focuses on selecting a subset of unlabeled data that, when labeled and used for training, produces the best-performing model (Kirsch and Gal 2022). AL focuses on model performance vs. number of labels, and it requires exactly one question per label, e.g., it requires N binary questions to annotate N samples. Our work differs from active learning in that it seeks to annotate N samples using *fewer than* N binary questions. By asking binary questions about groups of samples, our method reduces the total number of annotations needed to label the entire dataset accurately, thus enhancing annotation efficiency in a way that active learning does not address.

Theoretical Results

The data in binary classification comprise the set of observations $X = (x_1, \dots, x_N)$ and their associated ground truth labeling $y = (y_1, \dots, y_N) \in \{0,1\}^N$. Our objective is to infer y by asking binary questions to an annotator who has full knowledge of the vector y. As shown in (Kulkarni, Mitter, and Tsitsiklis 1993), all binary questions boil down to checking if the ground truth y is among a given set of vectors or not. In other words, a general binary question Q is formally defined by a set of L binary vectors, each with length N (i.e. $Q = \{q_1, \dots, q_L\}$ where $q_i \in \{0,1\}^N$ for $i = 1, \dots, L$), and its answer is yes when such set contains y (i.e. $a(Q) = \mathbf{1}_{y \in Q} \in \{0,1\}$).

An annotation process aims to determine the true y by looking at the answers $a_1(Q_1), \dots, a_k(Q_K)$, as each answer a_i provides information about whether y belongs to Q_i or to its complement Q_i^c. A question Q is equivalent to its complement Q^c since they provide the same information when answered. Assume without loss of generality that all answers are positive (else we swap Q and Q^c), then the state of knowledge S_k of the true labeling is summarized by the intersection of the k questions asked so far $S_k = \cap_{i \le k} Q_i$. A useful question makes the set of possible labelings smaller (i.e. $|S_{k+1}| < |S_k|$). We take useful questions to be $Q_{k+1} \subset S_k$, for which the next state is either Q_{k+1} or Q_{k+1}^c. An annotation process is a sequence of useful questions that deterministically depends on the answers and possibly on a probability distribution $P(Y = y)$ and for which the final state yields the true labeling (i.e. $S_K = \{y\}$). In particular, the traditional annotation process (one component y_i at a time) is a binary search for y where $|S_k|$ is halved at each step. Note that an annotation process can be seen as a binary tree where nodes are states of knowledge, branches are useful questions and their complements, and leaves are singleton sets each containing an N-long binary vector. For

[1] To bridge both notations, we take the concept of interest c to be the set of samples with positive ground-truth labels.

this tree, the children of a node are the pair (Q, Q^c). We now provide the link with coding theory.

Theorem 1. *Let $y \in \{0,1\}^N$ be the unknown ground-truth labeling and $P(Y = y)$ a probability distribution assigned to it. The problem of annotating y with the minimum expected number of binary questions is equivalent to finding an optimal prefix-free encoding of $\{0,1\}^N$ under $P(Y = y)$, and the Q&A sequence corresponds to the decoding process of this encoding.*

Proof. In coding theory nomenclature, $\{0,1\}^N$ is our alphabet and $P(Y)$ is the probability for each symbol, where y is the symbol of interest. By the Huffman coding theorem, there exists a (bijective) prefix-free encoding process $C : \{0,1\}^N \to \{0,1\}^*$ such that $\mathbb{E}[|C(Y)|]$ is minimal (Huffman 1952). The encoding process can be seen as a binary tree representation of C. An annotation process arises when interpreting the parent nodes of the binary tree as states S_k and their children as Q_{k+1} and Q^c_{k+1}. The decoding process uniquely determines y after $|C(y)|$ nodes or questions. In other words, Huffman decoding can serve as an annotation process. Now, we need to show that no annotation process can do better than Huffman encoding. To do this, we must show that any annotation process can be transformed into a prefix-free encoding process, as Huffman encoding is optimal among those. Given an annotation process, construct its decision tree taking Q^c at the left branches (i.e. $a = 1$ goes towards the right). The codeword for y is the path from root to leaf in this tree, i.e. $C(y) = [a_1(Q_1), \ldots, a_K(Q_K)]$. This encoding uniquely determines y (because any leaf does) and is prefix-free (else it would not be deterministic). Thus it is a valid encoding method, and the minimum expected number of questions $\mathbb{E}[K]$ must be at least the minimum expected code length of any prefix-free encoding. □

This result shows that the optimal questions are given by Huffman decoding. Note that multiple different optimal annotation processes might exist; for instance, any binary search is optimal when $P(Y = y)$ is uniform. Furthermore, we do not consider errors in the annotation process, which would require error correction methods. The theorem shows that the annotation problem is essentially an optimal encoding one. Now, we describe two exact methods to build the tree of the optimal annotation processes.

Huffman Encoding

The source coding theorem establishes that the minimum expected length achievable by a lossless compressor is given by the entropy $H(Y) = -\sum_{y \in Y} P(Y = y) \log(P(Y = y))$ (Shannon 2001). Huffman encoding (Huffman 1952) is a lossless encoding method that is optimal for the case of single-symbol encoding (this is proven via an exchangeability argument and induction). Note that optimally encoding a sequence of symbols is a different problem often solved with arithmetic codes. Huffman encoding achieves the entropy bound when the probabilities assigned to every symbol are dyadic, i.e. of the form $1/2^i$. Furthermore, one can prove that $H(Y) \leq Q_{\text{huffman}} \leq H(Y) + 1$ (Goemans 2015), which we observe experimentally in Figure 1. Theorem 1 guarantees that using Huffman as the annotation method we will arrive at the ground truth labeling y in the minimum expected number of questions K. We provide the algorithm in the Appendix. However, the computational complexity for building the tree is $m \log m$, where m is the number of symbols in the alphabet. In our case, $m = 2^N$. Unfortunately, the exponential complexity on the number of samples N becomes unmanageable for $N > 20$.

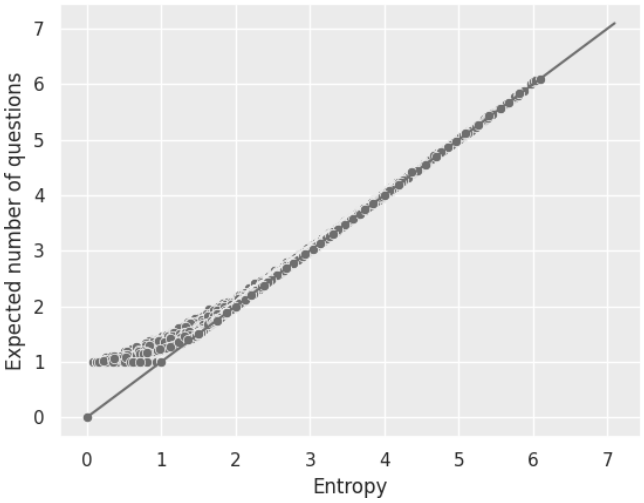


Figure 1: Given the Huffman tree and the probabilities of the leaves, we can compute the exact expected number of questions after each node. As expected, the expected number of questions gets very close to the entropy lower bound. Experiment over the synthetic dataset (a) of Section . The straight line is the identity function.

Exhaustive Tree Search and DP

Let us find the optimal cost function $J^\star$. This function assigns to each state the minimum possible cost achieved by a policy (i.e. annotation process) starting from such state. A state S_k comprises the set of remaining possible labelings and is a node in a tree, similarly to the Huffman case. But this tree is not binary, as any question $Q_{k+1} \subset S_k$ might be asked. We are interested in the annotation process that chooses the best (Q_{k+1}, Q^c_{k+1}), i.e. in the optimal policy, which can be seen as a binary tree. Assume the dyadic case where the Huffman policy reaches the entropy lower bound, i.e. $E[K] = H(Y)$. This lower bound reached by the Huffman policy is the minimum possible cost achieved by a policy from a given state is $H(Y|Y \in S_k)$, which is exactly the definition of the optimal cost function $J^\star$. Given $J^\star$ the derivation of the optimal policy is simple: we choose the action (Q_{k+1}, Q^c_{k+1}) that minimizes the expected cost $J^\star$ on the next state. In other words, given a state S_k we consider all possible questions $Q \subset S_k$ and choose

$$\begin{aligned} \arg\min_{Q_{k+1}} \big[& J^\star(Q_{k+1}) P(a(Q_{k+1}) = 1) \\ & + J^\star(Q^c_{k+1}) P(a(Q_{k+1}) = 0) \big] , \end{aligned} \tag{1}$$

where the answer $a(Q)$ defines the next state. Note that to compute $H(Y|Y \in S_k)$, we need to update the prob-

abilities $P(Y = y)$ using the Bayes rule to restrict them to the state S_k, which simply accounts for a renormalization. This method gives us a procedure that is optimal and greedy, in contrast to Huffman, which requires building a tree. The computational complexity is given by that of the entropy computation multiplied by the number of possible questions. The number of possible questions in the first step is the cardinality of the set of all possible subsets of binary vectors of length N; this is the cardinality of the power set $|2^Y| = 2^{2^N}$. This exponential-exponential computational cost is way higher than the exponential of Huffman and obviously unfeasible.

Practical Setting

The optimal solutions described above have two issues that make them useless. The first one is computational complexity, which is (at least) exponential on the dataset size. The second one is the type of binary questions, which are general and concern the whole dataset. The user should check every time if any of the proposed full dataset annotations are correct, which is unrealistic. Our proposed method solves these two issues. It leverages that lookahead minimization guarantees improvement over the base heuristic and drastically reduces the branching factor of the tree (originally 2^{2^N}). Another quantity of interest that will be used in what follows is the size $|S|$ of the state S, which represents the number of labelings that were not yet discarded and serves as an optional heuristic cost function to be minimized.

Reducing the Computational Cost

To reduce the branching factor of 2^{2^N} to something reasonable, we use a handful of heuristics. Each one of them takes us further away from optimality but improves efficiency, practicality, or user experience. We note that if the cardinality of a question Q is $n = |Q|$, the total number of possible questions can be computed by counting how many questions there are of each size n. This is $|\mathcal{Q}| = \sum_{n=1}^{2^N} \binom{2^N}{n} = 2^{2^N}$. The heuristics are:

Early Stopping: Call Q_n the best question of size n. Assume that the expected cost of asking Q_n as a function of n has only one local minimum. We look for it starting from $n = 1$ and increasing n until the cost increases at step $n = E$. This involves checking fewer cardinalities, thus reducing the branching factor to $\sum_{n=1}^{E} \binom{2^N}{n}$, where $E \leq N$.

Select the Most Likely: Although not generally true, we consider that the best question of size n is given by the n most likely labelings of those that remain available in S. In other words, we consider the most likely set of size n to be the best guess. This criterion is optimal when the cost function is $|S|$, but it is not necessarily optimal when the cost function is the entropy. If the labelings are sorted by probability, this heuristic reduces the branching factor to $\sum_{n=1}^{E} 1 = E$, as finding Q_n is immediate, else it takes $2^N \log(2^N) E$.

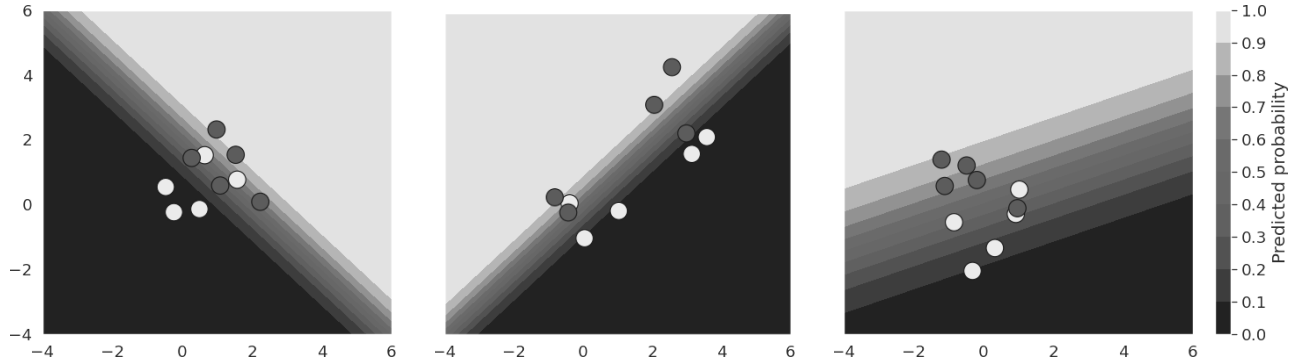
Note that the sorting can be made one single time only if the distribution $P(Y = y)$ stays the same throughout the annotation process. We now introduce two more heuristics that are not only motivated by the computational complexity issue but also by user experience considerations and algorithmic simplicity:

Question as Single Guess: This heuristic targets user experience: the space of questions is now restricted to the (most likely) guesses involving m observations $x_1, \ldots, x_m$. This means that **the method will choose some set of m observations, pseudo-label them, and ask the annotator if the pseudo-labeling is correct**. Note that $m = 1$ means that Q contains half of the vectors, i.e. $n = 2^{N-m}$. We adapt the *early stopping* (swap n by m above) and *select most likely* heuristics, that together cause our search to be of complexity $N \log(N) E$: we sort the probabilities for each label entry y_i and then evaluate the E first questions by increasing m.

Don't Give Up: This heuristic is introduced to simplify state management, which becomes contrived when guesses over vector entries are used. *Not giving up* means that when an incorrect guess of size m has been made, the next guess should be the same incorrect guess but without its most uncertain prediction. When the second guess is correct, we can deduce the labels for all the entries in the first guess. Otherwise, the second guess is incorrect, and we apply the heuristic again. This reduces the branching factor to 1 when an incorrect guess has been made.

Lookahead Minimization: The heuristics introduced above can successfully reduce the number of possible questions to be considered at each step. However, optimality was lost when reducing the question space. But using lookahead minimization, i.e. simulating a few steps ahead and only then evaluating the states using the chosen proxy cost function, even if it is not optimal, is guaranteed to improve over the heuristic generated by the proxy cost function (Bertsekas 2022). In the limit of maximum lookahead, the terminal states are reached, and the exact expected number of questions is computed, thereby achieving the optimal method respecting the above heuristics.

Summary and Parameters: In a nutshell, our method (dubbed IA) is lookahead minimization over the tree of states and yes/no questions. Some parameters (in italics) control the algorithm. The algorithm (full details in Appendix B) computes the cost of a state either by using the *proxy cost function* (see below) or by considering questions and answers (expansion) and computing the cost of the children states. The core of the algorithm is deciding which states are expanded. The selection of the state to be expanded is done according to the priority, where the priority depends on the "probability" of reaching the state and the "probability" of asking a question is a softmax over question costs with a given *temperature* (although question selection is always greedy). The questions are guesses with different sizes m, with a special treatment for $m = 1$. In fact, for $m = 1$, we allow to choose between *random sampling* or *entropy-based sampling*. Even though many other active learning algorithms exist (Zhan et al. 2022; Kirsch and Gal 2022), simple entropy-based active learning is an excellent baseline (Beck et al. 2021). To allow for more control over the branching factor and the user experience, a parameter

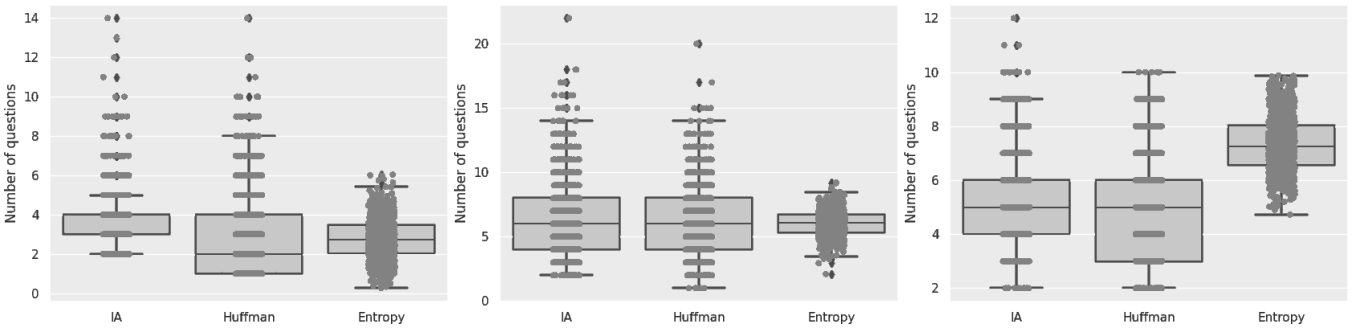


(a) Samples from two Gaussians. The predictor $f(X) = P(Y|X)$ obtained from Bayes' rule is perfect.

(b) Samples from two Gaussians, labels follow a logistic regression given by $f(X) = P(Y|X)$ (which is independent of $P(X)$).

(c) Data following (Guyon et al. 2006). The predictor f is a logistic regressor trained on the data and $f(X) \neq P(Y|X)$.

Figure 2: Example sample of three two-dimensional synthetic binary classification problems. The background color corresponds to the probabilities predicted by f. The color of the points corresponds to the ground truth labeling.



(a) (b) (c)

Figure 3: Number of questions for the synthetic problems (ref. Figure 2) for each method (IA, Huffman) and the entropy of the data as estimated by the predictor.

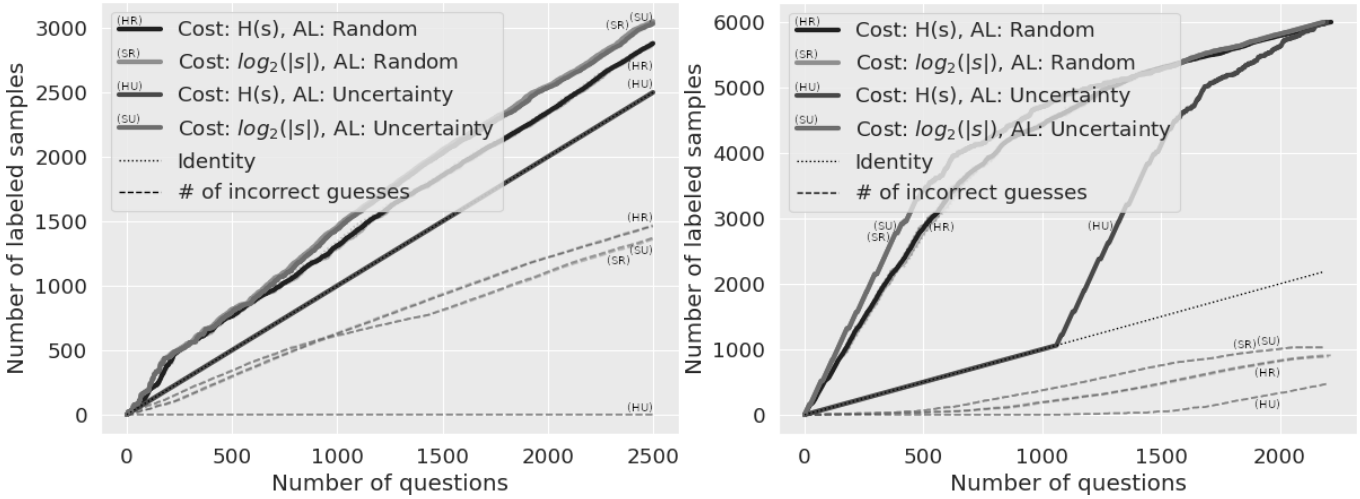
that sets the *maximum allowed guess size* m is added. Another parameter of our method is the *maximum node expansions* made at each step that roughly controls the computational cost of the tree search. Lastly, a *proxy cost function* is used, which is either the *entropy* $H(s)$ or the log state size $\log_2(|s|)$ (roughly the number of points left to annotate). Both functions are adapted for the case in which an incorrect guess is part of the state by using conditional entropy and counting concepts. After each state expansion, all parents are updated, which means that we must propagate the new estimate of the state's cost up the tree. Of course, if the probabilities given by $P(Y = y)$ do not change, then the relevant branch of the tree is kept in memory after each question, thereby allowing subsequent expansions to further enlarge the tree. We refer the reader to the supplementary material for the full code and algorithm details.

Estimating P: Assuming knowledge of $P(Y = y)$ is unrealistic. However, the distribution can be learned. A predictor f can be either pretrained or learned online. When calibrated, it estimates $f(x) \approx P(Y = y|x)$. In the practical scenario, we incrementally discover specific labels, but wrong guesses might happen. For these cases, the following method allows learning from mistakes. Consider a guess $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_m)$. The estimated probability of the guess $\hat{y}$ being correct is given by $g(\hat{y}\text{ ok}) = \prod_i f(x_i)^{\hat{y}_i}(1 - f(x_i))^{1-\hat{y}_i}$. Minimizing the binary cross entropy between $g(\hat{y}\text{ ok})$ and the negative answer $a = 0$ allows us to integrate the knowledge of wrong guesses into the weights of f.

Experiments

Here, we summarize our main results. The Appendix includes full numerical results, experimental details, and extra figures.

Synthetic: We generate three two-dimensional toy problem variations with dataset size $N = 10$ for 1000 different seeds. The variations are depicted in Figure 2 and detailed in Appendix C. In Figure 3, one observes that the proposed IA method is very close to the optimal given by Huffman ((a),(b), and (c)), which validates our implementation. For the cases where f is exactly $P(Y|X)$ the results are close to the entropy lower bound ((a) and (b)). For the last problem (c) we observe that the entropy computed by the predictor



(a) CIFAR10 (Krizhevsky, Hinton et al. 2009)

(b) SVHN (Netzer et al. 2011)

Figure 4: Number of annotations vs. number of questions for four IA variants on the first 6000 examples of each dataset and leveraging a fixed pretrained model.

that is fitted to the data is not a lower bound. However, look-ahead-minimizing it is highly effective.

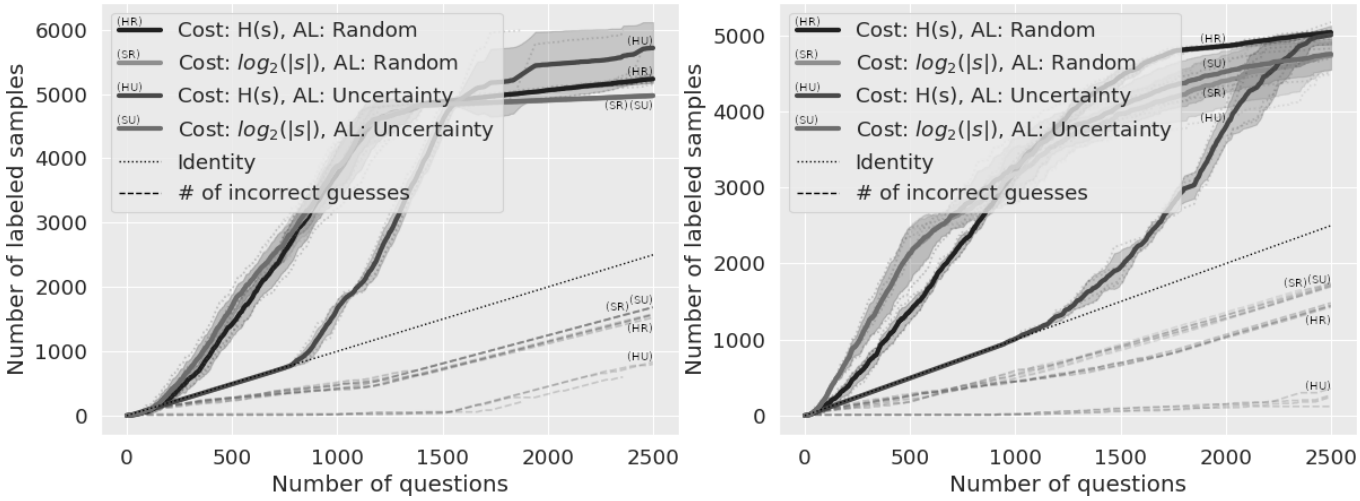
Real With Pretrained f: The IA method is evaluated over the first 6000 samples of the CIFAR10 (Krizhevsky, Hinton et al. 2009) and SVHN (Netzer et al. 2011) datasets, where all labels with indices greater than four are considered positive. These datasets are expensive to run with an online predictor and a fixed predictor trained on the rest of the data is provided to the IA. These oracles have accuracies of 0.67 and 0.93 respectively. Figure 4 shows the number of annotated samples vs. the number of binary questions posed (up to 2500). For CIFAR10, the maximum reduction in the number of questions required to annotate 2500 images is 20%, which is not bad considering that the predictor is not much better than random. In contrast, for SVHN, all methods manage to annotate the full 6000 samples with less than 2500 questions, showing how effective a good predictor can be when annotating data. We note that the number of incorrect guesses with $m > 1$ is permanently increasing in most cases, indicating non-trivial strategies. Furthermore, when using the entropy of the state $H(s)$ as a proxy cost function and offering the most uncertain sample as the potential guess for $m = 1$, the IA method sticks to point-by-point annotation for more than 1000 examples. This variant then switches to annotating the most obvious samples at a high annotation rate.

Real With Online f: Then, we turn our attention to the more realistic case of annotating when a trained predictor is not available. In this case, the predictor is trained from scratch and online using the labels as they become available. This is the case (dubbed ALIA) that combines the related tasks of active learning (training a predictor that selects samples to label) and quick annotation (using a predictor to annotate more efficiently) into human-in-the-loop learning with Q&A as the interaction paradigm. This ALIA process yields both annotated data and a trained predictor. For this experiment, we run ALIA over MNIST (LeCun et al. 1998) and Fashion MNIST (Xiao, Rasul, and Vollgraf 2017) (see Figure 5). Annotation cost is reduced by 70% for both datasets. As expected, after most easy samples are annotated, the annotation slows down. This change is very abrupt for MNIST, and we hypothesize that it is due to an uncalibrated predictor.

Discussion

On Speed and the Scalability to Larger Datasets: While we have not experimentally assessed large datasets, we can scale up by dividing a large dataset into manageable chunks. Moreover, by predicting the probabilities of these subsets of data once in a while, training and predicting asynchronously in the background, and tuning the computational cost of the tree search, the human-in-the-loop method can run in real-time.

Measuring Cost as the Number of Answers: The final cost is usually money or time, including development. Measuring cost in this way means running an annotation campaign; therefore, it is not a cheap way to measure. The alternative is to run simulations. Here, the cost is usually associated with the type of interaction, e.g. clicks for segmentation. But there is no universal way of interacting. For binary classification, annotating a dataset always implies answering binary questions. The added advantage of measuring cost as the number of questions is that, even if imperfectly, binary questioning can extend to most other tasks. For instance, in interactive image segmentation, one can binary search the position of the click on the image and ask for its label or simply ask for the label of a randomly chosen click. Another advantage of cost as binary answers is its atomicity: it is the smallest possible user input. The last advantage is its relation to information theory. Extensibility, atomicity, and



(a) MNIST (LeCun et al. 1998)

(b) FMNIST (Xiao, Rasul, and Vollgraf 2017)

Figure 5: Number of annotations vs. number of questions for four ALIA variants on the first 6000 examples of each dataset starting from a randomly initialized predictor. Corresponding tables in Appendix E.

theory-friendliness are unique characteristics of measuring cost as the number of binary answers. Finally, previous work also considers this metric.

Applicability: The method proposed is useful under the assumption that the cost per answer is constant. However, this is not always the case in practice. The method's effectiveness is strongly related to the ability of the annotator to do batch processing, which might depend on the problem. For instance, humans are good at detecting visual anomalies, but they might need to read each piece of text individually before deciding on the correctness of pseudo-labeling. A favorable case might be audio classification, where many short clips can be combined. We suggest checking the validity of the batch processing hypothesis using psychometric experiments before developing annotation tools.

The Method Outside Binary Classification: To extend the method to more complex settings, one should frame the new problem as binary Q&A. This is possible in many cases; for instance, in multiclass classification, one-hot encoding formats the ground truth as a binary matrix. Then, the method proposed here can be applied. In fact, (Hu et al. 2020) is a particular case of our method, where questions are restricted to be of a specific kind. The core principle behind our method is that given nonuniform probabilities for the ground truth, questions that, in expectation, annotate more entries (or reduce the entropy the most) should be preferred. This principle is generally useful once the problem is framed as binary Q&A.

Towards Full Human-in-the-Loop Learning: When human-in-the-loop learning, active learning asks for a single uncertain label, while our method asks for many less uncertain labels. Active learning has better sample efficiency, while our method provides more samples in fewer queries. The tension between annotating relevant samples and quickly annotating was not evident up to this work. When minimizing the entropy, our method does the balancing automatically, with a period of active learning before quickly annotating (see HU in Figure 5). But in this case, the goal was still quick annotation. Future work could formalize the human-in-the-loop learning problem as *maximizing performance vs. number of binary questions*, thereby resolving the tension by unifying both approaches.

Conclusion

Our work takes a step towards understanding and reducing annotation costs, and future work should explore the interplay between quick annotation and active learning, study imbalanced datasets, and assess the impact of miscalibration. Several results were presented: *First* this work formalized the problem of quick annotation as a binary Q&A game between an algorithm and an annotator and defined the annotation cost as the number of binary answers needed to annotate the dataset. *Second*, we showed a link to coding theory, which carries important theoretical consequences, namely that the best conceivable annotation method is Huffman decoding and that it significantly reduces the annotation cost even when the cost of annotating one sample is fixed. *Third*, given the prohibitive computational cost of Huffman, we developed a practical method that approximates exhaustive tree search, integrates active learning, and is computationally efficient. *Fourth*, we have shown how the probability of the labels can be approximated by a predictor that learns from annotations and also from mistakes and that such a predictor can be trained online. *Lastly,* the method is shown to work well on synthetic data, where it is close to optimal, and in real data, where it reduces the annotation cost by up to 70% even if no trained predictor is initially available. We hope our work serves as a first step toward a theory of annotation.

Acknowledgements

This work was supported by a CIFRE grant from ANRT. It was also partially financed by ANII Uruguay. Centre Borelli is also with Université Paris Cité, SSA and INSERM.

References

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Beck, N.; Sivasubramanian, D.; Dani, A.; Ramakrishnan, G.; and Iyer, R. 2021. Effective Evaluation of Deep Active Learning on Image Classification Tasks. arXiv:2106.15324.

Bertsekas, D. 2012. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific.

Bertsekas, D. 2021. *Rollout, policy iteration, and distributed reinforcement learning*. Athena Scientific.

Bertsekas, D. 2022. Rollout Algorithms and Approximate Dynamic Programming for Bayesian Optimization and Sequential Estimation. *arXiv preprint arXiv:2212.07998*.

Beyer, L.; Hénaff, O. J.; Kolesnikov, A.; Zhai, X.; and Oord, A. v. d. 2020. Are we done with imagenet? *arXiv preprint arXiv:2006.07159*.

Bhambri, S.; Bhattacharjee, A.; and Bertsekas, D. 2022. Reinforcement Learning Methods for Wordle: A POMDP/Adaptive Control Approach. *arXiv preprint arXiv:2211.10298*.

Goemans, M. 2015. Huffman Codes.

Guyon, I.; Gunn, S.; Hur, A. B.; and Dror, G. 2006. Design and analysis of the NIPS2003 challenge. *Feature extraction: foundations and applications*, 237–263.

Hu, H.; Xie, L.; Du, Z.; Hong, R.; and Tian, Q. 2020. One-bit Supervision for Image Classification. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 501–511. Curran Associates, Inc.

Huffman, D. A. 1952. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9): 1098–1101.

Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A. C.; Lo, W.-Y.; et al. 2023. Segment anything. *arXiv preprint arXiv:2304.02643*.

Kirsch, A.; and Gal, Y. 2022. Unifying Approaches in Active Learning and Active Sampling via Fisher Information and Information-Theoretic Quantities. *Transactions on Machine Learning Research*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kulkarni, S. R.; Mitter, S. K.; and Tsitsiklis, J. N. 1993. Active learning using arbitrary binary valued queries. *Machine Learning*, 11: 23–35.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, Q.; Cho, J.; Bansal, M.; and Niethammer, M. 2024. Rethinking Interactive Image Segmentation with Low Latency High Quality and Diverse Prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3773–3782.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Assran, M.; Ballas, N.; Galuba, W.; Howes, R.; Huang, P.-Y.; Li, S.-W.; Misra, I.; Rabbat, M.; Sharma, V.; Synnaeve, G.; Xu, H.; Jegou, H.; Mairal, J.; Labatut, P.; Joulin, A.; and Bojanowski, P. 2023. DINOv2: Learning Robust Visual Features without Supervision. arXiv:2304.07193.

Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C.; Ma, T.; Khedr, H.; Rädle, R.; Rolland, C.; Gustafson, L.; et al. 2024. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*.

Shannon, C. E. 2001. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1): 3–55.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Xie, B.; Yuan, L.; Li, S.; Liu, C. H.; and Cheng, X. 2022. Towards fewer annotations: Active learning via region impurity and prediction uncertainty for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8068–8078.

Yang, L.; Hanneke, S.; and Carbonell, J. G. 2010. Bayesian Active Learning Using Arbitrary Binary Valued Queries. In *ALT*, 50–58. Springer.

Zhan, X.; Wang, Q.; Huang, K.-h.; Xiong, H.; Dou, D.; and Chan, A. B. 2022. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*.