Some Label Efficient Learning Results

David Helmbold
Computer Science Department
UC Santa Cruz
Santa Cruz, CA 95064
dph@cse.ucsc.edu

Sandra Panizza*
Computer Science Department
UC Santa Cruz
Santa Cruz, CA 95064
panizza@cse.ucsc.edu

Abstract

We investigate the value of labels in a simple version of the standard on-line prediction model (the "experts" setting). We present algorithms and adversary arguments defining tradeoffs between the number of mistakes made and the number of labels that the learner requests. One version of this question can be viewed as a family of games whose value is given by a complicated recurrence. Although our attempts to find a closed form for this recurrence have been unsuccessful, we show how an algorithm can efficiently compute its value, enabling it to perform optimally.

1 INTRODUCTION

Everyone knows that a picture is worth a thousand words, but how much is a label worth to a learning algorithm? We investigate this question in a simple version of the standard on-line prediction model (the "experts" setting). We present algorithms and adversary arguments that define interesting tradeoffs between the number of mistakes made and the number of labels that the learner requests. One version of this question turns into a family of games whose value is given by a complicated recurrence. Although our attempts to find a closed form for this recurrence have been unsuccessful, we show how an algorithm can efficiently compute its value, enabling it to perform optimally.

In the on-line prediction model the learner (or predictor) must predict, one by one, the labels $y_1, \ldots, y_T$ of an unknown sequence of instance/label pairs $\mathbf{s} = \langle x_1, y_1 \rangle, \ldots, \langle x_T, y_T \rangle$. Each instance x_t of the sequence $\mathbf{s}$ can be interpreted as a set of features to which the label y_t is associated. For example, in a text classification problem x_t might indicate those key phrases occuring in a particular document the learner is asked to classify and $y_t \in \{0, 1\}$ is "1" if the document belongs to the class of relevant documents and "0" otherwise. Formally, the learning process consists of a series of trials numbered from 1 to T. At the beginning of each trial t the learner receives the instance x_t and, based on x_t and on the past instance/label pairs, produces a prediction $\hat{y}_t \in \{0, 1\}$ for y_t. In the standard on-line model the label y_t is received at the end of the trial and is used by the learner to update its internal state, and (hopefully) improve its future predictions.

The learner's goal is usually to minimize the number of prediction mistakes made over all T trials on an adversarially generated sequence of instance/label pairs. Although this problem is difficult in general, if the learner is given a (finite) set of models and the sequence is close to that generated by one of the models, then certain performance bounds can be achieved [Vov90, LW94, CBFH+94]. This is sometimes referred to as the "experts" setting, since the models can be viewed as "experts" providing "advice" to the algorithm.

In this setting, the learner (or "master algorithm") gives the current instance x_t to all of the experts. Each expert E_i $(i = 1, \ldots N)$ informs the master algorithm of its prediction on the current instance. The algorithm then combines the experts' advice to produce its own prediction $\hat{y}_t$. At the end of the trial, the label y_t is shared with all the experts and both the experts and the master algorithm are scored based on the accuracy of their predictions. The master's goal is to perform nearly as well as the best expert on the particular sequence that was actually observed. For example, one simple case occurs when one of the experts perfectly predicts all of the labels in the sequence s, and the learner must quickly learn to follow the predictions of the good expert in order to minimize the number of prediction mistakes. One solution for this simple case is the well-known Halving algorithm that, on each trial, simply predicts the same way as the majority of those experts that have never made a mistake.

In the standard on-line prediction model, the complexity of a learning problem is measured in terms of the number of prediction mistakes made by the best (not necessarily effi-

*Supported by a foreign study fellowship from the University of Milan.

Permission to make digital/hard copies of all or part of this material for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copyright is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires specific permission and/or fee

COLT 97 Nashville, Tennesee, USA
Copyright 1997 ACM 0-89791-891-6/97/7..$3.50

cient) on-line prediction algorithm, and the goal is to develop efficient master algorithms whose complexity is "close" to that of the optimal algorithm. However the successful design of practical algorithms often requires the conservation of critical resources in addition to minimizing the number of prediction mistakes made. In many applications, such as speech recognition or text categorization, the labels used by the master algorithm can be a scarce resource. Although many instances may be available (such as documents from the web or speech samples from broadcast radio or TV), obtaining the correct classification can be difficult and/or expensive. It is often the case that collecting unlabeled instances can be easily automated, while determining the correct labeling of the instances requires expensive human expertise.

Thus it is desirable to study on-line prediction algorithms that make few mistakes while asking for few labels during the learning process. In this framework, learning becomes an interactive process in which the algorithm chooses on each trial whether or not to request the label, instead of automatically receiving every label.

Note that if the learner does not request a label then it is receiving less information than it would in the standard on-line model, and thus can be expected to make poorer predictions in the future. This leads to an explicit trade-off between the number of mistakes made and the number of labels requested. Furthermore, if an algorithm hopes to optimize this tradeoff, then its strategy for requesting labels must be chosen carefully.

The purpose of this work is to explore the trade-offs between the numbers of labels requested and mistakes made, and to find appropriate strategies for requesting labels within the worst case framework adopted by the standard on-line prediction model. In doing so, we present a natural variant of the on-line prediction model, which we call the *label-efficient* prediction model. In this model the learner, at the end of each trial, chooses whether or not to receive the correct classification of the instance. The performance of a learning algorithm now includes both the number of labels requested by the algorithm as well as the number of prediction mistakes made. Since the trade-off between the number of labels asked and the number of mistakes made is explicit in the label-efficient model, different prediction problems can be defined depending on how the trade-off is weighted. For some of these problems we present, through a game theoretic analysis, an optimal yet efficient prediction strategy.

We concentrate on the simple setting where the predictions and outcomes are boolean and the master algorithm knows in advance that one of the experts predicts perfectly on the sequence $\mathbf{s}$. The binomial weighting technique described by Cesa-Bianchi *et al.* [CBFHW96] is an efficient way to build a perfect expert whenever some expert makes a small number of mistakes. Even in this simple setting the tradeoff between mistakes made and labels requested is quite subtle. Furthermore, a thorough understanding of this simple setting is required before one can hope to prove interesting results in more complex or realistic settings, such as when the predictions or outcomes are continuous and/or the target is a convex combination of the experts.

Although much work has been done in the on-line learning model, the most closely related work to that presented here is the *Apple Tasting* model presented by Helmbold, Littlestone, and Long [HLL92]. In this Apple Tasting model the learner receives the correct label for the current instance if and only if it makes the prediction $\hat{y}_t = 1$. Intuitively, the model represents the situation where the learner is attempting to identify (and eat) good apples, while avoiding the bad ones. A prediction $\hat{y}_t = 1$ is associated with eating the apple (and thus obtaining confirmation of its goodness or badness), and a prediction $\hat{y}_t = 0$ represents avoiding the current apple (whose goodness or badness remains unknown). The measure of performance is simply the number of prediction mistakes made, or the number of tasty apples passed up plus bad apples eaten. Depending on the concept class involved, the number of mistakes made by an optimal apple tasting algorithm typically grows as $\sqrt{T}$, where T is the number of predictions made. This model involves an implicit exploration/exploitation tradeoff – good learners must occasionally predict "1" to gain information even when the label is likely to be "0". In contrast to the apple tasting model, the label-efficient prediction model studied here makes the exploration/exploitation tradeoff explicit by counting the labels requested separately from the number of mistakes made by the algorithm.

Atlas *et al.* [ACL+90] consider *selective sampling*, a closely related model to ours where the learner (in their case a neural network) requests the labels only of those points in its region of uncertainty. They present a method for detecting this region of uncertainty as well as experimental results. One very interesting selective sampling algorithm is the "query by committee" algorithm [SOS92, FSST93], which ensures that the average information provided by each requested label is bounded from below by a constant. This allows the generalization error to decrease exponentially in the number of labels, as opposed to the slower inverse-power law that applies when the labeled examples are selected at random. The query by committee algorithm works by filtering randomly drawn examples using a committee of Gibbs algorithms, and only those examples on which the committee has maximum disagreement are passed through the filter.

There are two main differences between the selective sampling model and the label efficient prediction model presented here. First, in the selective sampling model the instances are drawn randomly from a fixed distribution, whereas the instances are generated adversarially in our label efficient model. Second, the label efficient model is an on-line model where the algorithm must make predictions (and run the risk of mistakes) on those instance where the label is not requested. In contrast, the selective sampling models are concerned with the generalization error rather

than the performance of the algorithm on the sequences of instances. Although there are conversions between on-line and batch algorithms [Lit89, CBFH+93, HW95], the selective sampling/query by committee algorithms can often wait for a more "perfectly informative" example than algorithms in the label efficient prediction model.

Since the label efficient prediction model requests the labels of instances, it (and the selective sampling model) may appear similar to learning models with membership queries (such as [Val84, Ang88]). There is, however, a very important difference – the membership query models usually allow the algorithm to *construct* the instances for their queries, whereas the label efficient prediction model (as well as selective sampling) are allowed to request the labels only of those instances appearing in their input.

In the next section we describe the label-efficient prediction model in more detail and Section 3 describes and analyzes some simple adversaries and algorithms. In Section 4 we give a game theoretic interpretation of the label efficient prediction model and describe how analyzing the game leads to an efficient optimal algorithm.

2 NOTATION AND MODEL

As stated above, learning in the label-efficient prediction model proceeds in a series of trials numbered from 1 to T. In a little more detail, on each trial t:

1. the adversary chooses the pair (x_t, y_t)
2. the learner obtains the predictions or advice of the N experts on x_t
3. the learner formulates its own prediction $\hat{y}_t$
4. the learner then decides whether or not to ask for the label, and the value y_t is given to the learner only when it is asked for.

The ideal behavior of the algorithm is to always form correct predictions $\hat{y}_t = y_t$ while not asking for any labels. However, this ideal behavior is rarely achievable. In the standard on-line setting it is natural to charge the algorithm one mistake for each trial where $\hat{y}_t \neq y_t$. However, in the label-efficient prediction model it may be reasonable for the algorithm to delay acting until the requested label is available (and the appropriate action is known). In this case, one should only charge the algorithm a mistake when it both predicts incorrectly and fails to ask for the label.

In this paper we define the *total error* as the number of trials on which $y_t \neq \hat{y}_t$, and the number of *mistakes* as the number of trials on which both $y_t \neq \hat{y}_t$ and the learner fails to ask for the label. Thus the mistakes made by the algorithm will not include the "bad guesses" when the algorithm also requests the label, but the total error does include these bad guesses. These quantities are trivially related by the following inequalities:

$$\text{mistakes} \leq \text{total error} \leq \text{mistakes} + \text{labels requested}.$$

Although our analysis emphasizes mistakes, our results can be phrased in terms of total error using the above inequalities. The important point is that the number of labels requested and number of mistakes (or total error) charged to the algorithm form a two-dimensional loss. The goal of the algorithm is to minimize this two-dimensional loss and a different version of the label-efficient prediction problem results when a different emphasis is placed on the numbers of labels requested and mistakes made.

We will consider only adversaries that produce sequences of pairs $\langle (x_t, y_t) \rangle_{1 \leq t \leq T}$ where at least one of the N experts predicts correctly on every trial. We call these perfectly predicting experts *consistent*. As mentioned in the introduction, standard techniques can be used to create a consistent expert whenever at least one of the experts makes only a small number of mistakes. At each trial, the current *version space* is the set of experts that have predicted perfectly on the previous trials where the algorithm has asked for the label. If every expert in the version space makes the same prediction, then the outcome is known to be that value. Thus if only a single expert remains in the version space then the algorithm has learned the rule labeling the instances and need not make any further mistakes nor ask any further labels.

This basic setting is parameterized by the number of experts (N) and number of trials (T). For each N, T pair, we are interested in the pairs (L, M) such that there is an algorithm which on every sequence of length T and any set of N experts (where at least one of the experts is consistent) the algorithm requests at most L labels and makes at most M mistakes.

In general, the algorithm's decision whether or not to ask for the label and the choice of prediction can be randomized. However, we will often find it convenient to restrict the algorithm so that it predicts in the same way as the majority of the experts in the current version space. We call such algorithms *sensible*. Note that sensible algorithms produce the same predictions as those generated by the well-known halving algorithm.

One intriguing property of our model is that sensible algorithms are optimal. In most other on-line settings, it is beneficial to predict with a little randomness unless the vast majority of the version space gives the same advice. However, we show in Section 4.5 that an algorithm that predicts randomly only when the version space is evenly split (or it is out of labels) has the optimal (expected) mistake bound.

3 PROPERTIES OF THE LABEL EFFICIENT PREDICTION MODEL

There are several trivial performance bounds which one can show for the label-efficient prediction model. The halving algorithm in this setting works as follows: predict with majority of the version space and request the label unless the entire version space gives the same advice. Although this algorithm never makes a mistake (by our definition) since it requests the label whenever the outcome is unknown, it can

have total error as large as $\log N$. Furthermore, the halving algorithm requests as many as $N-1$ labels. In fact, an adversary can postpone the halving algorithm's errors to the end (when the version space is smaller) to create sequences of trials where the halving algorithm both has total error m and requests $N+m-2^m$ labels.

This halving algorithm is essentially optimal in the following sense: any algorithm whose expected number of mistakes is bounded must request a number of labels approaching N. This can be seen by considering an adversary which on every trial has one expert from the version space predict "1" and the rest predict "0". The adversary will repeat this set of expert predictions until the algorithm requests the label. If the algorithm eventually requests the label with probability one, then the adversary uses the outcome "0" so that only one expert is eliminated from the version space. By repeating this process on the reduced version space, the algorithm eventually requests $N-1$ labels. On the other hand, if there is some positive probability that the algorithm never requests the label then the adversary uses the label "1" and the expected number of mistakes made by the algorithm becomes unbounded. Thus this adversary forces any sensible algorithm to either make an expected number of mistakes that is unbounded or to eventually request $N-1$ labels.

This indicates that interesting bounds in the label-efficient prediction model must depend on T, the length of the sequence to be predicted. Note that a similar dependency occurs in the apple tasting model, where the mistake bounds typically grow as $\sqrt{T}$. The following theorem quantifies this intuition.

Theorem 1 *If the expected number of labels requested by a sensible algorithm is bounded by $L \leq N/3$, then the algorithm can be forced to make an expected number of mistakes that is at least $2T/(9L)$.*

Proof. We prove the Theorem by presenting an adversary strategy forcing $2T/(9L)$ expected mistakes on any sensible algorithm. To avoid floors and ceilings in notation, we will assume that T is a multiple of $3L$. The adversary strategy focuses on the first $3L$ of the N experts which we call the relevant experts. The adversary divides the T trials into $3L$ groups of $T/(3L)$ trials each. The predictions of the experts depend only on which group the trial is in: in group i expert E_i ($i = 1, \ldots, 3L$) always predicts "1" while all other experts predict "0". Thus on the first $T/(3L)$ trials, the first expert, E_1, predicts "1" while all other experts predict "0". On the next $T/(3L)$ trials, expert E_2 predicts "1" while all other experts predict "0", and so on, with expert E_{3L} predicting "1" on the last $T/(3L)$ trials.

At the beginning of the learning process the adversary chooses (uniformly at random) one of the relevant experts to be the consistent expert. The labels for the sequence of trials are then set to the same value as the predictions of the chosen expert. Thus the chosen expert remains consistent throughout the process.

Since the expected number of labels requested by the algorithm is at most L, there is at most a 1/3 chance that the algorithm asks for a label on any of the $T/(3L)$ trials where the consistent expert predicts 1. Since the majority of current version space predicts "0" on these trials, so will the algorithm. Thus the expected number of mistakes made by the sensible algorithm is at least $\frac{2}{3}T/(3L) = 2T/(9L)$. □

This lower bound gives us our first estimate on the value of a label. As the number of labels used by the algorithm increases, the lower bound goes down harmonically (as $1/L$).

The lower bound of Theorem 1 is tight to within a $\lg N$ factor. Consider the simple algorithm which requests a label with probability $p = L/T$ whenever the predictions of the version space are not unanimous. The following version space argument shows that this algorithm expects to make at most $(T-L)(\lg N)/L$ mistakes. The version space initially contains all N experts. The size of the version space is reduced by at least a factor of two on each trial where the algorithm both predicts incorrectly and requests the label. Therefore, the version space will shrink by (at least) a factor of two after an expected $1/p - 1$ mistakes (recall that no mistakes are charged on trials where the algorithm requests the label). Since the version space can shrink at most $\lg N$ times in this way before it is reduced to a single expert, the algorithm makes an expected number of mistakes M bounded by

$$M \leq (\lg N)/p - \lg N = (T-L)(\lg N)/L \qquad (1)$$

mistakes. Observe that the expected number of labels requested by this algorithm is trivially bounded by $pT = L$.

We can fix the number of mistakes rather then the number of labels requested. When $p = (\lg N)/(M + \lg N)$, the expected number of mistakes made by the algorithm is at most M while the expected number of labels requested is at most $T(\lg N)/(M + \lg N)$.

The following theorem shows that one can do slightly better by asking for the label with a probability that depends on how evenly the predictions of the version space are split between "0" and "1". Intuitively, if most of the version space predicts in the same way, then the algorithm should only rarely ask for the label. On the other hand, if the experts predicting "1" and the experts predicting "0" split the version space evenly then the algorithm should ask for the label more aggressively.

Theorem 2 *For all $M \geq 0$, there is an algorithm which expects to make at most M mistakes and expects to ask for at most $L = \frac{T \lg N}{M \lg(1/r^*) + \lg N}$ labels, where r^* is the solution to $N(1-r)^T = r^M$.*

Proof Sketch. The algorithm asks for the label with probability $p(r) = \frac{\lg N}{\lg N + M \lg(1/r)}$, where $0 \leq r \leq 1/2$ is the fraction of the version space predicting in the minority. This is the smallest p guaranteeing that, when the minority is always correct, the version space shrinks rapidly enough so that the expected number of mistakes is bounded by M. It turns out

that the best adversary strategy is to use the same split r on every trial. Thus the expected number of labels requested is bounded by both $T\,p(r) = \frac{T \lg N}{\lg N + M \lg(1/r)}$ and the number of labels required to exhaust the version space when the majority of the version space is always correct. This number of labels required to reduce the version space to a single expert is $\frac{\lg N}{\lg(1/(1-r))}$. These two bounds are equal (and thus their minimum is maximized) when $N(1-r)^T = r^M$. Calling the solution of this equation r^*, and substituting it into the first bound completes the proof. □

We can relate Theorem 2 to the previous bounds as follows. By solving the equation in Theorem 2 for M, we see that when the expected number of labels is bounded by L, the algorithm makes an expected number of mistakes M bounded by

$$M \leq (T - L)(\lg N)/(L \lg(1/r^*)), \qquad (2)$$

and thus shaves a factor of $\lg(1/r^*)$ off the earlier bound (1). Furthermore, if $r^* \approx 1/N$, then mistake bound (2) grows as T/L, just like the lower bound in Theorem 1. A little algebra shows that $r^* = 1/N$ when $(M+1)/T = (\lg \frac{N}{N-1})/\lg N \approx 1/(N \ln N)$, or $T \approx (M+1)N \ln N$. Under these conditions, the lower bound of Theorem 1 is within a constant factor of the upper bound (2).

4 ON-LINE PREDICTION USING A BOUNDED NUMBER OF LABELS

This section contains the main result of the paper, a game theoretic analysis yielding an optimal learning algorithm when the number of requests for labels is limited by a fixed bound. As before, we restrict our analysis to the simple case when all the predictions are Boolean and the learning algorithm knows in advance that one of the experts will be consistent with the sequence s (i.e. it predicts perfectly on the sequence).

In this section we extend our definition of "sensible." Whereas before we insisted that sensible algorithms always predict in the same way as the majority of the version space, we now allow sensible algorithms to predict with unbiased coin flips in two special situations. First, our earlier definition of sensible did not specify how the algorithm predicted when the predictions of the current version space are evenly split. We adopt the natural convention that the algorithm predicts with an unbiased coin flip in this case. Second, the restriction that the algorithm never request more than a fixed number of labels leads to a degenerate situation where the algorithm is at the mercy of the adversary. Consider what happens when the algorithm is no longer able to ask for labels and three (or more) experts remain in the version space. Now the adversary can easily force a mistake on every trial when the algorithm must blindly predict as the majority of the version space. To counter this, we also allow sensible algorithms to predict with an unbiased coin flip once the quota of label requests has been exhausted.

Surprisingly, this very limited randomization of predictions coupled with the randomized choice of when to ask for a label is sufficient to construct an optimal algorithm (as will be shown in Section 4.5). This is in marked contrast to most other on-line settings where the best expected mistake bounds are achieved by algorithms that randomize their predictions on most trials.

Throughout this section we view the label-efficient prediction model as a game in which the learner plays against an opponent or adversary who dynamically generates both the (binary) experts' advice and the (binary) outcomes to be predicted. The game is parameterized by the number N of experts, the number T of trials played, and the number L of labels the algorithm is allowed to request during the game. We use "$\mathcal{G}(N,T,L)$" to denote the game with parameters N, T, and L. Formally, given the triple (N,T,L), the $\mathcal{G}(N,T,L)$ game consists of T time steps or trials. On each trial the protocol of Section 2 is used, and the value of the game is the (expected) number of mistakes made by the learner. The goal of the learner in this game is to minimize the number of mistakes charged to him/her over the T trials, while the adversary's goal is to maximize it. Recall that the learner is **not** charged a mistake when the label is requested.

During any $\mathcal{G}(N,T,L)$ game, we say that the *state* of the game at the beginning of a trial is the triple (n,t,ℓ) where n is the size of the current version space, t is the number of remaining trials and ℓ is the remaining number of labels that the algorithm can request.

Since we are interested in a worst-case setting where the experts' advice and outcomes can be thought of as being generated by an adversary, any *deterministic* strategy for requesting the labels would fail very badly against an adversary who assigns the wrong classification to the majority of the current experts when the label is not requested by the algorithm and the correct classification otherwise. Thus, in order to achieve good performances the learning algorithm must use a randomized selection strategy. In particular we consider the learning algorithm that uses a probability function p for deciding, on each trial, whether or not to ask for the label. That is, we assume that on each trial the learner flips a coin with some bias $p \in [0,1]$ and then requests the label if and only if the outcome of the coin toss is "heads." Note that the algorithms in the previous section uses either a fixed bias, or a bias that depended on the fraction of the version space predicting "1." In this section we will examine an algorithm whose probability of requesting the label depends not only on the current predictions of the version space, but also on the state of the game.

4.1 VALUE OF THE $\mathcal{G}(N,T,L)$ GAME

Recall that for any $N \geq 1$, $L \geq 0$ and $T \geq 0$, the goal of the adversary in the $\mathcal{G}(N,T,L)$ game is to maximize the expected number of mistakes made by the prediction algorithm. In this context, we will show that the minimax value of the $\mathcal{G}(N,T,L)$ game is the function $v(N,T,L)$ defined

Boundary Conditions

$$B1: \ v(1,t,\ell) = 0; \qquad B2: \ v(n,t,0) = \frac{t}{2} \ \forall n \geq 2;$$

$$B3: \ v(n,t,\ell) = 0 \text{ when } t \leq l.$$

Recursive Formulation

For any $n \geq 2$, for any $t \geq 2$, and for any $\ell \geq 1$

$$v(n,t,\ell) = \max_{\{i \in \mathbf{N} \,:\, i \leq \lfloor \frac{n}{2} \rfloor\}} \{v_i(n,t,\ell)\} \tag{3}$$

where if $i < n/2$,

$$\begin{aligned} v_i(n,t,\ell) = \inf_p \max\{ & \\ & p\,v(n-i,t-1,\ell-1) + (1-p)v(n,t-1,\ell), \\ & p\,v(i,t-1,\ell-1) + (1-p)[1 + v(n,t-1,\ell)]\} \end{aligned} \tag{4}$$

and when $i = n/2$,

$$\begin{aligned} v_i(n,t,\ell) = \inf_p \{ & \\ & p\,v(\frac{n}{2},t-1,\ell-1) + (1-p)[\frac{1}{2} + v(n,t-1,\ell)]\} \end{aligned} \tag{5}$$

Figure 1: Value of the $\mathcal{G}(n,t,\ell)$ Game.

recursively in Figure 1. An efficient method for computing $v(N,T,L)$ is given in Section 4.2.

The parameters of the function v are interpreted as follows. The integer n denotes the size of the current version space (i.e. the number of experts that have predicted perfectly on those previous trials where the algorithm requested the label), the integer t is the number of remaining trials, and the integer ℓ is the remaining number of labels that can be requested by the algorithm. We now justify that the recurrence for $v(n,t,\ell)$ given in Figure 1 correctly represents the value of the $\mathcal{G}(n,t,\ell)$ game.

We start with the boundary conditions. For condition B1, it is easy to see that when only a single expert remains in the version space, the algorithm has learned which expert is consistent, and thus won't make any further mistakes. Boundary condition B2 follows immediately by observing that when $n \geq 2$ and the algorithm is left with 0 labels the best strategy for the algorithm is to predict with an unbiased bit, and thus it expects to make $1/2$ of a mistake on each of the t remaining trials. Finally, B3 follows by noting that if $t \leq \ell$ then the algorithm can request the label on all of the remaining trials. Because mistakes are not charged when the algorithm requests the label, the algorithm will make no mistakes and the value of the game is 0.

We need some additional notation before justifying the recursion. On each trial the experts' advice partitions the current version space V into V_0 and V_1 where $V_{y'}$ is the subset of V predicting y' on the current trial. If the current version space contains $n \geq 2$ experts then we say that the experts' advice produce an $(i, n-i)$ *split* where $i = \min\{|V_0|, |V_1|\}$. Since the experts' advice is selected by the adversary, the choice of $(i, n-i)$ split, for $i \in \{1,2,\ldots,\lfloor \frac{n}{2} \rfloor\}$ is also controlled by the adversary. In this respect, $v_i(n,t,\ell)$ given in figure 1 denotes the value of the $\mathcal{G}(n,t,\ell)$ game when the adversary selects an $(i, n-i)$ split on the first trial.

When n is odd the recursion in figure 1 simplifies to

$$\begin{aligned} v(n,t,\ell) = \max_{\{i \in \mathbf{N} : i < \frac{n}{2}\}} \inf_p \max\{ & \\ & p\,v(n-i,t-1,\ell-1) + (1-p)v(n,t-1,\ell), \\ & p\,v(i,t-1,\ell-1) + (1-p)\,[1 + v(n,t-1,\ell)]\} \end{aligned} \tag{6}$$

To explain equation (6), two cases must be considered: when the majority of the experts in the current version space predict the correct value of the label, and when the majority of the experts predict the wrong value. For the sake of explanation, let us assume that an $(i, n-i)$ split is selected by the adversary for the first trial, where $i \in \{1,2,\ldots,\lfloor \frac{n}{2} \rfloor\}$. When the majority of the experts (and thus the algorithm) predict correctly on the first trial, no mistake will be charged to the algorithm. The state of the game after this trial becomes either $(n-i, t-1, \ell-1)$ if the algorithm asks for the label, or $(n, t-1, \ell)$ if the algorithm does not request the label. Since the algorithm asks for the label with probability p, the expected value is

$$p\,v(n-i,t-1,\ell-1) + (1-p)\,v(n,t-1,\ell).$$

When the majority of the experts, and thus the algorithm, predict incorrectly on the first trial, the algorithm will be charged a mistake only when the label is not requested. In this case the expected value of the game is

$$p\,v(i,t-1,\ell-1) + (1-p)\,[1 + v(n,t-1,\ell)].$$

Now, (6) immediately follows by observing that the adversary, in order to maximize the loss of the algorithm, has to choose the value $i \in \{1,2,\ldots,\lfloor \frac{n}{2} \rfloor\}$ maximizing the right-hand-side of (6).

The n even case differs only in that on an $(\frac{n}{2}, \frac{n}{2})$ split the learning algorithm should always predict randomly, thus incurring an expected number of mistakes of $1/2$ rather than 1 on each trial where the label is not requested. This is taken into account by the expansion for $v_{n/2}(n,t,\ell)$ in equation (5) of figure 1.

4.2 SOME EXPLOITABLE STRUCTURE OF THE GAME

Before presenting an optimal prediction strategy for the $\mathcal{G}(N,T,L)$ game, some results about the value $v(N,T,L)$ of the game are needed. These results enables us to simplify the recursion equation presented in figure 1. Furthermore, they provide the theoretical justification for an efficient algorithm solving this game.

We first give two simple technical Lemmas.

Lemma 3 *For any number $n \geq 1$ of experts and for any number $\ell \geq 0$ of labels, the value $v(n, \ell+1, \ell)$ of the $\mathcal{G}(n, \ell+1, \ell)$ game is bounded by*

$$v(n, \ell+1, \ell) \leq \frac{1}{2}. \tag{7}$$

Proof. Consider the algorithm which requests the label on the first ℓ trials and predicts with a random bit on the last trial. This algorithm makes an expected number of mistakes equal to $1/2$ (since no mistakes are charged when the algorithm requests the label), proving the lemma. □

Lemma 4 *For any $n \geq 1$, for any $\ell \geq 0$ and for any $t \geq 0$, the following inequality holds,*

$$v(n, t+1, \ell) \leq \frac{1}{2} + v(n, t, \ell).$$

Proof. The thesis follows by considering the algorithm that plays the $\mathcal{G}(n, t, \ell)$ game for the first t trials, and then either requests the label or predicts with a random bit on the last trial. □

Next we provide two Lemmas which compute the best choice for the probability p used by the algorithm for requesting the label, as a function of the split used by the adversary on the first trial.

Lemma 5 *For any $n \geq 2$, $\ell \geq 1$, $t \geq \ell + 1$ and for any integer $i < \frac{n}{2}$, the minimizing p of $v_i(n, t, \ell)$ is either $p = 1$ or $p = 1/(v(n-i, t-1, \ell-1) - v(i, t-1, \ell-1) + 1)$. In other words, for $i < \frac{n}{2}$*

$$v_i(n, t, \ell) = \min\{\, v(n-i, t-1, \ell-1),$$

$$\frac{(v(n-i, t-1, \ell-1) - v(i, t-1, \ell-1))v(n, t-1, \ell)}{v(n-i, t-1, \ell-1) - v(i, t-1, \ell-1) + 1} +$$

$$+ \frac{v(n-i, t-1, \ell-1)}{v(n-i, t-1, \ell-1) - v(i, t-1, \ell-1) + 1}\}. \tag{8}$$

Proof Sketch. By analyzing the derivatives with respect to p of the two terms in the max. □

Note that Lemma 5 does not apply when the adversary chooses a $(n/2, n/2)$ split. However, for even splits the best strategy for the algorithm is to request the label. This makes intuitive sense since this is the only case where the requested label is guaranteed to provide a full bit of information.

Lemma 6 *For any $n \geq 1$, for any number of labels $\ell \geq 1$ and for any $d \geq 1$,*

$$v_n(2n, \ell + d, \ell) = v(n, \ell + d - 1, \ell - 1) \tag{9}$$

Proof By induction on d. For completeness, the proof is provided in the appendix. □

An interesting Corollary of Lemma 6 which enables us to simplify the recursion presented in figure 1, is the following.

Corollary 7 *For any $n \geq 2$, for any $t \geq 2$, and for any $\ell \geq 1$, we have*

$$v(n, t, \ell) = \max_{i \in \{1, 2, \ldots, \lfloor \frac{n}{2} \rfloor\}} \{v_i'(n, t, \ell), \} \tag{10}$$

where

$$v_i'(n, t, \ell) = \inf_p \max\{$$
$$p\, v(n-i, t-1, \ell-1) + (1-p)\, v(n, t-1, \ell),$$
$$p\, v(i, t-1, \ell-1) + (1-p)\,[1 + v(n, t-1, \ell)]\}$$

is the value of the $\mathcal{G}(n, t, \ell)$ game when the adversary makes an $(i, n-i)$ split on the first trial.

Proof. The proof follows directly by noting that when n is even and $i = \frac{n}{2}$ is substituted in equation (4) of figure 1 we obtain the term $p\, v(\frac{n}{2}, t-1, \ell-1) + (1-p)(1 + v(n, t-1, \ell))$ which is minimized for $p = 1$. Moreover, for $i = n/2$ the probability functions defined in Lemma 5 both reduce to 1. By Lemma 6, this implies that for n even the value $v_{\frac{n}{2}}(n, t, \ell)$ (equation (5)) is equal to the value given by equation (4) after setting $i = n/2$. This concludes the proof. □

Intuitively, Corollary 7 says that if there are labels remaining and the adversary uses a $(n/2, n/2)$ split, then an optimal algorithm can predict either randomly or deterministically on that trial. This is because the optimal algorithm will request the label, and in our model the learner is not charged a mistake when the label is requested. Since the recursion in figure 1 and in Corollary 7 are equivalent (i.e. they both compute the same function), in the following we will use the simpler definition given in Corollary 7.

Remark:
It is not difficult to see that when the value of the game is computed as in Corollary 7, the probabilities given in Lemma 5 are also the minimizing p for the value $v_i'(n, t, \ell)$ of the game. An important special case is when $t = \ell + 1$. In this case we have that for any integer $i \leq \lfloor n/2 \rfloor$,

$$v_i'(n, \ell+1, \ell) = \frac{v(n-i, \ell, \ell-1)}{(v(n-i, \ell, \ell-1) - v(i, \ell, \ell-1) + 1)}. \tag{11}$$

Although, our attempts to find a closed form for the value of the game have been unsuccessful, the next Theorem indicates that the game has exploitable structure allowing the $v(n, t, \ell)$ values to be computed efficiently.

Theorem 8 *For any $n \geq 1$, for any $\ell \geq 0$ and $d \geq 1$, we have*

$$v(n, \ell + d, \ell) = d\, v(n, \ell + 1, \ell). \tag{12}$$

Proof. The theorem is proved by using a double induction on d and ℓ. The proof is provided in the appendix. □

The important feature of Theorem 8 is that it removes the time parameter t from the recurrence. By pre-computing the values $v(n, \ell+1, \ell)$, an algorithm can easily obtain the probability p from Lemma 5 that minimizes the expected number of mistakes made. This is the basis for the algorithm presented in section 4.4

4.3 A LOWER BOUND ON THE VALUE OF THE GAME

We next prove a lower bound on the value $v(n,t,\ell)$ of the $\mathcal{G}(n,t,\ell)$ game in the label efficient prediction model we have been considering. We begin with the following definition.

Definition 9 *For any $n \geq 2$ and for any $t \geq 1$, we define $f(n,t)$ to be the function*

$$f(n,t) = \begin{cases} 1/2 & \text{if } n \geq 2^t \\ 1/3 & \text{if } 2^{t-1} < n < 2^t \\ \frac{1}{s(n,t)+3} & \text{if } n \leq 2^{t-1} \end{cases}$$

where $s(n,t)$ is the smallest s satisfying $2^{t-s-1} - (t-s) < n - t < 2^{t-s} - (t-s)$.

We now present the main result of this section.

Theorem 10 *For any $n \geq 1$, for any $t \geq 0$ and for any $\ell \geq 0$, the value $v(n,t,\ell)$ of the $\mathcal{G}(n,t,\ell)$ game can be bounded by*

$$v(n,t,\ell) \geq Low(n,t,\ell)$$

where

$$Low(n,t,\ell) = \begin{cases} 0 & \text{if } t \leq \ell \text{ or } n = 1 \\ t/2 & \text{if } \ell = 0 \text{ and } n > 1 \\ (t-\ell) f(n,\ell+1) & \text{otherwise} \end{cases}$$

and $f(n,\ell+1)$ is the function given in definition 9.

Note that, when $t \leq \ell$ or $n = 1$ or ($\ell = 0$ and $n > 1$), the lower bound given in Theorem 10 trivially holds since in these cases $v(n,t,\ell) = Low(n,t,\ell)$. Thus, the bound need only be proved for $n \geq 2$, $t > \ell$ and $\ell \geq 1$. Lemma 11 below shows that for the special case $t = \ell + 1$ the function $f(n,\ell+1)$ is a lower bound on the value $v(n,\ell+1,\ell)$ of the $\mathcal{G}(n,\ell+1,\ell)$ game. The bound of Theorem 10 then follows immediately since by Theorem 8 of section 4.2, for $t > \ell$ we have $v(n,t,\ell) = (t-\ell)\, v(n,\ell+1,\ell)$.

Lemma 11 *For any $n \geq 2$ and for any $\ell \geq 1$*

$$v(n,\ell+1,\ell) \geq f(n,\ell+1). \tag{13}$$

Proof. By definition (see Corollary 7 of section 4.2) we have $v(n,\ell+1,\ell) = \max_{\{i \in \mathbf{N} \,:\, i \leq \lfloor \frac{n}{2} \rfloor\}} \{v'_i(n,\ell+1,\ell)\}$ where

$$\begin{aligned} & v'_i(n,\ell+1,\ell) \qquad (14) \\ &= \inf_p \max\{p\, v(n-i,\ell,\ell-1) + (1-p)\, v(n,\ell,\ell), \\ &\qquad\qquad p\, v(i,\ell,\ell-1) + (1-p)(1+v(n,\ell,\ell))\} \\ &= \inf_p \max\{pv(n-i,\ell,\ell-1), p(v(i,\ell,\ell-1)-1)+1\} \\ &= \frac{v(n-i,\ell,\ell-1)}{v(n-i,\ell,\ell-1) - v(i,\ell,\ell-1) + 1}, \qquad (15) \end{aligned}$$

where the second equality follows from the fact that $v(n,\ell,\ell) = 0$ and the last equality by noting that the best choice for p is $p = 1/(v(n-i,\ell,\ell-1) - v(i,\ell,\ell-1) + 1)$. Using the fact that for any $j \leq \lfloor n/2 \rfloor$

$$v(n,\ell+1,\ell) \geq v'_j(n,\ell+1,\ell), \tag{16}$$

different lower bounds on the value $v(n,\ell+1,\ell)$ of the game can be derived by assuming different splits on the first trial. Now, to prove (13) three cases must be considered depending on the relationship between n and ℓ.

We start by analyzing the case $n \geq 2^{\ell+1}$. Since $n \geq 2^{\ell+1}$ and only ℓ labels can be asked during the whole game, the best strategy for the algorithm is to always ask for the label, hence incurring no mistakes during the first ℓ trials. On the last trial, where no further label can be asked, the algorithm incurs no mistakes if the version space contains only a single expert, and an expected number of mistakes equal to $1/2$ if the version space contains at least two experts. The best strategy for the adversary is to choose a sequence of splits such that the version space on the last trial contains at least two experts. It is easy to see that if the adversary uses (roughly) even splits on all of the trials when $n \geq 2^{\ell+1}$ then $v(n,\ell+1,\ell) = 1/2 = f(n,\ell+1)$.

Next, we consider the case $2^\ell < n < 2^{\ell+1}$. Using inequality (16) with $j = n - 2^\ell$ we obtain

$$\begin{aligned} v(n,\ell+1,\ell) &\geq v_{n-2^\ell}(n,\ell+1,\ell) \\ &= \frac{v(2^\ell,\ell,\ell-1)}{v(2^\ell,\ell,\ell-1) - v(n-2^\ell,\ell,\ell-1) + 1} \\ &= \frac{1}{3 - 2\, v(n-2^\ell,\ell,\ell-1)} \end{aligned}$$

where the first equality follows from equation (15), and the last equality from the fact that $v(2^\ell,\ell,\ell-1) = 1/2$ (see the analysis for $n \geq 2^{\ell+1}$). Then, the lower bound $f(n,\ell+1) = 1/3$ immediately follows by noting that $v(n-2^\ell,\ell,\ell-1) \geq 0$.

Finally, we consider $n \leq 2^\ell$. Proceeding similarly to the previous case but with $j = 1$, we obtain

$$\begin{aligned} v(n,\ell+1,\ell) &\geq v_1(n,\ell+1,\ell) \\ &= \frac{v(n-1,\ell,\ell-1)}{v(n-1,\ell,\ell-1) - v(1,\ell,\ell-1) + 1} \\ &= \frac{1}{1 + \frac{1}{v(n-1,\ell,\ell-1)}} \qquad (17) \end{aligned}$$

where equality (17) follows by noting that $v(1,\ell,\ell-1) = 0$.

Now, for any pair of integer (n,t) where $n \leq 2^{t-1}$ consider the succession rule whose ith term is $(n-i, t-i)$ for $i = 0, 1, \ldots, \min\{n-1, t-1\}$ and define $s(n,t) = i^*$ where $i^* = \arg\min_{i=0,1,\ldots,\min\{n-1,t-1\}} \{2^{t-i-1} < n-i < 2^{t-i}\}$. By setting $\Delta = n - t$ the succession can be written conveniently as $(\Delta + i, i)$ for $i = t, t-1, \ldots, 0$, where now $s(n,t) = t - i^*$ and $i^* = \arg\max_{i=t,t-1,\ldots,0} \{2^{i-1} - i < \Delta < 2^i - i\}$. It is not difficult to see that $s(n,t)$ is the smallest s satisfying $2^{t-s-1} - (t-s) < n - t < 2^{t-s} - (t-s)$.

Since for the first $s(n,\ell+1)$ trials we have that $n' \leq 2^{\ell'-1}$ where n' denotes the size of the version space on the

trial been played and ℓ' the number of remaining trials, by recursively applying the lower bound (17) to each of the values occuring in the right hand side of (17) we obtain

$$\begin{aligned} v(n,\ell+1,\ell) &\geq \frac{1}{s(n,\ell+1)+\frac{1}{v(n'',\ell''+1,\ell'')}} \\ &\geq \frac{1}{s(n,\ell+1)+3} \end{aligned} \quad (18)$$

where (18) follows by observing that $2^{\ell''} < n'' < 2^{\ell''+1}$ and that, from the analysis of the previous case, $v(n'', \ell'' + 1, \ell'') \geq 1/3$. This concludes the proof of the Lemma. □

Although upper bounding $v(n,t,\ell)$ has proven difficult, we believe that the lower bound given in Theorem 10 is quite tight, and conjecture that for n and t greater than ℓ, the value $v(n,t,\ell)$ is at most a factor of two greater than our lower bound.

4.4 AN OPTIMAL PREDICTION STRATEGY

We now give the optimal prediction strategy for the $\mathcal{G}(N,T,L)$ game described above, where N is the number of experts in the pool, T is the number of trials, and L is a bound on the number of labels that can be requested by the learner during the game. Recall that the state of the game at the beginning of a trial is the triple (n,t,ℓ) where n is the size of the current version space, t is the number of remaining trials, and ℓ is the remaining number of labels that the algorithm can request.

Prediction Strategy PS.

- Upon receiving the advice of the N experts the PS strategy predicts the same way as the experts if the version space is unanimous. Otherwise, it predicts with an unbiased random bit in the special cases when either $\ell = 0$ and $n > 1$ (no further labels can be asked) or the experts' advice splits the version space into two sets of exactly equal size. Normally, the PS strategy predicts in the same way as the majority of the experts in the current version space.
- If either $\ell = 0$ or the version space is unanimous then the PS strategy does not request the label, else it uses the values $v(n,t,\ell)$ of the $\mathcal{G}(n,t,\ell)$ to compute a probability of asking for the label as described below.

  - The algorithm asks for the label with probability $p = 1$ whenever $t \leq \ell$, $n \geq 2^t$, or when the predictions of the experts in the current version space are evenly split, otherwise
  - the algorithm exploits Lemma 5 by computing the probability function

$$p = \frac{1}{v(n-i,t-1,\ell-1) - v(i,t-1,\ell-1) + 1},$$

and asks for the label with probability p.

We introduce some notation before analyzing the performance of the prediction strategy PS. Given an adversary A for the $\mathcal{G}(n,t,\ell)$ game, we define $M_{PS,A}(n,t,\ell)$ to be the expected number of mistakes made by PS when it plays against adversary A in the $\mathcal{G}(n,t,\ell)$ game. We also define $M_{PS}(n,t,\ell)$ to be the supremum over all possible adversary strategies A of the expected number of mistakes incurred by the prediction strategy PS during the $\mathcal{G}(n,t,\ell)$ game, that is

$$M_{PS}(n,t,\ell) = \sup_A \{M_{PS,A}(n,t,\ell)\}.$$

The main Theorem concerning the prediction strategy PS is the following.

Theorem 12 *For any $N \geq 1$, for any $T \geq 0$ and for any $L \geq 0$, for the PS strategy we have*

$$M_{PS}(N,T,L) = v(N,T,L),$$

where $v(N,T,L)$ is the function satisfying the recurrence given in figure 1, or equivalently, in Corollary 7.

Proof. The Theorem is proved by induction on T. When $T = 0$ both sides are zero and the thesis holds. Now assume that the assertion is true for some $T - 1 \geq 0$, i.e. for any $N \geq 1$ and $L \geq 0$ we have $M_{PS}(N,T-1,L) = v(N,T-1,L)$. We will now show that $M_{PS}(N,T,L) = v(N,T,L)$ is also true. Assume that an $(i, N-i)$ split is chosen by the adversary on the first trial. Since PS asks for the label with probability $p = 1/(v(N-i,T-1,L-1) - v(i,T-1,L-1)+1)$, the maximum expected number of mistakes made by PS during the game is equal to $f_i(N,T,L)$ where

$$\begin{aligned} f_i(N,T,L) = \max\{ & \\ & pM_{PS}(N-i,T-1,L-1) \\ & \quad +(1-p)M_{PS}(N,T-1,L), \\ & pM_{PS}(i,T-1,L-1) \\ & \quad +(1-p)[M_{PS}(N,T-1,L)+1]\}. \end{aligned}$$

Since we have $M_{PS}(N-i,T-1,L-1) = v(N-i,T-1,L-1)$, $M_{PS}(i,T-1,L-1) = v(i,T-1,L-1)$, and $M_{PS}(N,T-1,L) = v(N,T-1,L)$ by the inductive hypothesis, we can substitute in the value used for p by PS and rewrite $f_i(N,T,L)$ as

$$\frac{(v(N-i,T-1,L-1) - v(i,T-1,L-1))v(N,T-1,L)}{v(N-i,T-1,L-1) - v(i,T-1,L-1)+1}$$

$$+\frac{v(N-i,T-1,L-1)}{v(N-i,T-1,L-1) - v(i,T-1,L-1)+1}.$$

The same techniques used to prove Theorem 8 (see Appendix) can be used to show that the maximum over $i \leq \lfloor N/2 \rfloor$ of the above expression is equal to $(T-L)v(N,L+1,L)$. Thus,

$$\begin{aligned} M_{PS}(N,T,L) &= \max_{\{i \leq \lfloor N/2 \rfloor\}} \{f_i(N,T,L)\} \\ &= (T-L)v(N,L+1,L) \\ &= v(N,T,L), \end{aligned}$$

concluding the proof of the Theorem. □

Note that an improved prediction strategy can be obtained if the strategy PS asks for the label with a probability $p = 1$ when 1 is the minimizing p of $v_i^t(n, t, \ell)$ and its usual p otherwise. Although this does not change the worst case expected mistake bound, it will slightly improve PS's performance against weak adversaries.

Theorem 12 shows that PS is an optimal strategy for the $\mathcal{G}(N, T, L)$ game. However, we have not yet shown that PS can be implemented efficiently. A straightforward implementation takes $O(TLN^2)$ time per prediction to compute the required $v(n, t, l)$ values (say, using dynamic programming to build a table of all $v(n, t, l)$ with $n \leq N$, $t \leq T$, and $\ell \leq L$). Although we can assume $L < N$ (else $v(N, T, L) = 0$, and the PS strategy will always ask for the label when the version space disagrees), the value of T need not be polynomial in N. Fortunately, Theorem 8 of section 4.2 shows that $v(n, t, \ell) = (t - \ell)v(n, \ell + 1, \ell)$. Therefore the algorithm need only compute a size N^2 table of $v(n, \ell + 1, \ell)$ values using dynamic programming. Since we know the best value of p, each entry can be computed in $O(N)$ time by maximizing over the split, and the entire table can be built in $O(N^3)$ time. Note that this table need be computed only once, and using the table the algorithm can compute its probability of asking for the label in constant time.

4.5 RANDOMIZED PREDICTION IN THE $G(N, T, L)$ GAME

Up until now we have considered sensible learners that predict in the same way as the majority of the experts in the current version space unless either the version space is evenly split or no more labels can be asked by the algorithm. In this section we show that learners which randomize their predictions are no better than the sensible PS learner of the previous section.

In the standard expert setting, it is well known that the expected number of mistakes when the learning algorithm is allowed to make randomized predictions is exactly half of the mistake bound when the learner is forced to make deterministic predictions. Thus, it is natural to ask whether randomized predictions also lead to improved prediction strategies in our more general label efficient model. Surprisingly, it turns out that randomization is not beneficial in our setting and that algorithms which predict randomly only when the version space is evenly split (or the algorithm is out of labels) have the optimal expected mistake bound.

We introduce some definitions before presenting this section's main result. We define $\mathcal{G}_R(n, t, \ell)$ to be the game described in section 4 when the prediction strategy is allowed to make randomized predictions during the game. This means that in the protocol of section 2, the learner may flip a biased coin to determine its prediction $\hat{y}$. We use the subscript R in the notation of the game to emphasize the use of randomized predictions. As with the $\mathcal{G}(n, t, \ell)$ game, the value $v_R(n, t, \ell)$ of the $\mathcal{G}_R(n, t, \ell)$ game is defined inductively as in figure 1 except that now the value $v_{R_i}(n, t, \ell)$ of the $\mathcal{G}_R(n, t, \ell)$ game when an $(i, n - i)$ split is used by the adversary on the first trial must also account for the randomized prediction. In particular, for any $i \leq \lfloor n/2 \rfloor$,

$$\begin{aligned} v_{R_i}(n, t, \ell) = \inf_{p,q} \max\{ & \\ & p\, v_R(n - i, t - 1, \ell - 1) \\ & \quad +(1 - p)[(1 - q) + v_R(n, t - 1, \ell)], \\ & p\, v_R(i, t - 1, \ell - 1) \\ & \quad +(1 - p)[q + v_R(n, t - 1, \ell)]\}. \end{aligned} \tag{19}$$

Here q represents the probability that the algorithm predicts in the same way as the majority of the current version space. Since in our model the algorithm is not charged a mistake when the label is requested, randomized predictions only matter when the algorithm doesn't ask for the label. In this case, equation (19) can be easily justified by noting that the algorithm incurs a mistake either when the majority is correct and the algorithm, with probability $1 - q$, predicts with the minority, or when the majority is wrong and with probability q the algorithm predicts with the majority. We now present the main result of this section.

Theorem 13 *For any $n \geq 1, t \geq 0$, and $\ell \geq 0$, we have*

$$v(n, t, \ell) = v_R(n, t, \ell) \tag{20}$$

where $v(n, t, \ell)$ and $v_R(n, t, \ell)$ are the values of the $\mathcal{G}(n, t, \ell)$ and the $\mathcal{G}_R(n, t, \ell)$ games respectively.

As a simple first Corollary, we see that the prediction strategy PS presented in section 4.4 has the optimal (expected) mistake bound in the label efficient randomized prediction model considered in this section.

Corollary 14 *For any $N \geq 1$, for any $T \geq 0$ and for any $L \geq 0$, for the PS strategy of section 4.4 we have*

$$M_{PS}(N, T, L) = v(N, T, L) = v_R(N, T, L)$$

where $v(N, T, L)$ and $v_R(N, T, L)$ are the values of the $\mathcal{G}(N, T, L)$ and $\mathcal{G}_R(N, T, L)$ games respectively.

To prove Theorem 13 we need the following technical Lemma.

Lemma 15 *For any $n \geq 2, \ell \geq 1, t \geq \ell + 1$, and for any integer $i \leq \lfloor n/2 \rfloor$, the minimizing (p, q) of $v_{R_i}(n, t, \ell)$ is $q = 1$ and either $p = 1$ or $p = 1/(v_R(n - i, t - 1, \ell - 1) - v_R(i, t - 1, \ell - 1) + 1)$.*

Proof. It is not difficult to see that the minimizing p of the max in the right hand side of (19) is either $p = 1$ or $p = p(q) = (2q - 1)/(v_R(n - i, t - 1, \ell - 1) - v_R(i, t - 1, \ell - 1) + 2q - 1)$. Since the algorithm is not charged a mistake when the label is requested, it follows that when $p = 1$ is the minimizing p of the right hand side of (19) then any q,

in particular $q = 1$, can be used. Now we consider the case $p = p(q)$.

For ease of notation, we let $a = v_R(n-i, t-1, \ell-1)$, $b = v_R(i, t-1, \ell-1)$ and $c = v_R(n, t-1, \ell)$. When $p = p(q)$ is substituted in the right hand side of (19) we obtain the function $f(a, b, c, q)$ where

$$f(a, b, c, q) = \frac{aq + ac - bc - b + bq}{a - 1 + 2q - b}.$$

Now finding the q minimizing $v_{R_i}(n, t, \ell)$ (and thus minimizing the expected number of mistakes) is equivalent to minimizing the function $f(a, b, c, q)$. Computing the derivative of $f(a, b, c, q)$ with respect to q we obtain

$$\frac{\partial f(a, b, c, q)}{\partial q} = \frac{a^2 - a + b - b^2 - 2ac + 2bc}{(a - 1 + 2q - b)^2}. \quad (21)$$

Since the sign of the derivative is independent of q, the minimizing q of $f(a, b, c, q)$ is at the boundary, either $q = 0$ or $q = 1$ depending on whether the derivative is positive or negative. Now we show that $\partial f(a, b, c, q)/(\partial q) \leq 0$ which in turn implies that $q = 1$ is the desired solution.

Since the denominator in the right hand side of (21) is always positive, it follows that $\partial f(a, b, c, q)/(\partial q) \leq 0$ iff $g(a, b, c) \leq 0$ where

$$g(a, b, c) = a^2 - a + b - b^2 - 2ac + 2bc.$$

Note that a, b, and c are all non-negative, and that $a \geq b$. If we set $a = b + d$, we can rewrite $g(a, b, c)$ conveniently as $g(a, b, c) = d(2b + d - 1 - 2c)$. It is now sufficient to show that $2b + d - 1 - 2c \leq 0$, or equivalently, that

$$2\,v_R(n, t-1, \ell) + 1 \geq v_R(n-i, t-1, \ell-1) + v_R(i, t-1, \ell-1). \quad (22)$$

To show that (22) holds we argue as follows. Since Lemma 4 of section 4.2 holds also for $v_R(n, t, \ell)$, we obtain $2\,v_R(n, t-1, \ell) + 1 \geq 2\,v_R(n, t, \ell)$. Furthermore, since for any $i \leq \lfloor n/2 \rfloor$, $v_R(n, t, \ell) \geq v_{R_i}(n, t, \ell)$ it follows that $2\,v_R(n, t-1, \ell) + 1 \geq 2\,v_{R_i}(n, t, \ell)$. Now to simplify the proof, we define

$$\begin{aligned} T_{Mc}(p, q) &= p\,v_R(n-i, t-1, \ell-1) \\ &\quad + (1-p)[(1-q) + v_R(n, t-1, \ell)]; \\ T_{Mw}(p, q) &= p\,v_R(i, t-1, \ell-1) \\ &\quad + (1-p)[q + v_R(n, t-1, \ell)]. \end{aligned}$$

Using the fact that $\inf_{p,q} \max\{A(p, q), B(p, q)\} \geq \inf_{p,q}[(A(p, q) + B(p, q))/2]$ the following chain of inequalities can be derived.

$$\begin{aligned} 2\,v_R(n, t-1, \ell) + 1 &\geq 2v_R(n, t, \ell) \geq 2v_{R_i}(n, t, \ell) \\ &= 2 \inf_{p,q} \max\{T_{Mc}(p, q), T_{Mw}(p, q)\} \\ &\geq 2 \inf_{p,q} \{(T_{Mc}(p, q) + T_{Mw}(p, q))/2\} \\ &= \inf_{p,q} \{T_{Mc}(p, q) + T_{Mw}(p, q)\} \quad (23) \end{aligned}$$

Since the right hand side of (23) is a linear function in p and q, the integers (p, q) minimizing (23) are in the set $p, q \in \{0, 1\}$. Noting that the right hand side of (23) is minimized when $p = 1$ and $q = 0$ or $q = 1$, which in both cases yields $2\,v_R(n, t-1, \ell) + 1 \geq v_R(n-i, t-1, \ell-1) + v_R(i, t-1, \ell-1)$, showing (22). This concludes the proof of the Lemma. □

Proof of Theorem 13.
To simplify the proof we define

$$\begin{aligned} f(p, q, i) &= \max\{p\,v_R(n-i, t-1, \ell-1) + (1-p) \\ &\qquad [(1-q) + v_R(n, t-1, \ell)], \\ &\qquad p\,v_R(i, t-1, \ell-1) + (1-p) \\ &\qquad [q + v_R(n, t-1, \ell)]\} \end{aligned}$$

The Theorem is proved by induction on t. The thesis trivially holds for $t = 0$. Now assume that the assertion holds for some $t-1 \geq 0$, i.e. for any $n \geq 1$ and $\ell \geq 0$ we have $v(n, t-1, \ell) = v_R(n, t-1, \ell)$. We need to show that $v(n, t, \ell) = v_R(n, t, \ell)$ is also true. Using Lemma 15 we have that

$$\begin{aligned} v_R(n, t, \ell) &= \max_i \inf_{p,q} \{f(p, q, i\} \\ &= \max_i \inf_p \{f(p, 1, i)\} \\ &= \max_i \{v_i(n, t, \ell)\} = v(n, t, \ell) \end{aligned}$$

where the third equality follows by the inductive hypothesis. This conclude the proof of the Theorem. □

5 CONCLUSIONS

In this paper we have presented a new on-line prediction model that explicitly represents labels as an important resource. Our results are phrased in the expert model, and we make the assumption that one of the experts makes perfect predictions. Since there are many settings in which the cost of labels is higher than that of (unlabeled) instances, it is important to understand how to do label-efficient learning.

Our first results show that if the sequence of trials is unbounded then an adversary can force any algorithm to either make an expected number of mistakes that is unbounded, or request a number of labels equal to the number of experts minus one. Despite this rather negative result, one can prove interesting bounds on the expected number of mistakes and expected label requests when there is a known bound T on the number of trials. We present a simple adversary showing that if L is the expected number of labels requested by the algorithm and L is at most one-third the number of experts, then the expected number of mistakes grows at least as fast as T/L. In a complementary result we give an algorithm showing that this lower bound is tight to within a log factor in general, and tight to within a constant in many cases.

Our main result is the analysis of the game that results when there is a fixed bound on the number of labels that the learner requests. The value of this game has an extremely complicated behavior. Although we currently have a lower

bound, we have been unable to prove a suitable upper bound on it. However the game does have structure that allows its value to be efficiently computed. This structure is exploited by our PS algorithm. The PS algorithm usually predicts deterministically, but does make random predictions when the version space is evenly split or it has run out of labels. Interestingly, this basically deterministic predictor is optimal even when randomized predictions are allowed in general.

One of the strengths of this model is that it separates the prediction from the label requests. In other on-line models the predictions of optimal algorithms must be randomized as a "hedge" against the adversary. In the label efficient model, the algorithm uses the probability of requesting the label as the hedge against the minority being correct. This allows an optimal algorithm to make (essentially) deterministic predictions, and seems a more appropriate way of randomizing against the adversary. It would be interesting to see if divorcing the label requests from the predictions has a similar effect in different settings. Examination of these problems could generate additional insight into crucial exploration/exploitation tradeoff issues.

Several interesting issues remain unresolved. Although a simple closed form for the value of the fixed-label game may not exist, it would be interesting to get an upper bound that approximately matches our (or an improved) lower bound. Would variations of the game (such as charging mistakes when a label is requested) be easier to evaluate?

Although worst-case bounds can be very illuminating, they are often overly pessimistic and the random generation of instances can give more realistic learning bounds. One important open problem is to define and analyzing a similar label efficient model when the instances are generated at random instead of adversarially.

On the other hand, we make the strong assumption that some expert always predicts the correct outcome. Another important way to generalize the model is to deal explicitly with inaccurate experts. The generalization to experts making real-valued predictions is a natural second step in that direction, as it gives the experts a way to express their confidence in their predictions. Work in this direction is underway.

References

[ACL+90] Les Atlas, David Cohn, Richard Ladner, M.A. El-Sharkawi, R.J. Marks II, M.E. Aggoune, and D.C. Park. training connectionist networks with queries and selective sampling. Number 2 in NIPS, 1990.

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, April 1988.

[CBFH+93] N. Cesa-Bianchi, Y. Freund, D. P. Helmbold, D. Haussler, R. E. Schapire, and M. K. Warmuth. How to use expert advice. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, pages 382–391. ACM Press, New York, NY, 1993. Expanded version in Univ. of Calif. Computer Research Lab TR UCSC-CRL-94-33, From Santa Cruz, CA.

[CBFH+94] N. Cesa-Bianchi, Y. Freund, D. Haussler, D. P. Helmbold, R. E. Schapire, and M. K. Warmuth. How to use expert advice. Technical Report UCSC-CRL-94-33, Univ. of Calif. Computer Research Lab, Santa Cruz, CA, 1994. Accepted subject to revision by *JACM*.

[CBFHW96] N. Cesa-Bianchi, Y. Freund, D.P. Helmbold, and M.K. Warmuth. On-line prediction and conversion strategies. *Machine Learning*, 25(1):71–110, October 1996.

[FSST93] Yoav Freund, H.S. Seung, E. Shamir, and N. Tishby. Accelerating learning using query by committee. NIPS, 1993.

[HLL92] D. P. Helmbold, N. Littlestone, and P. M. Long. Apple tasting and nearly one-sided learning. In *Proc. of the 33rd Symposium on the Foundations of Comp. Sci.*, pages 493–502. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[HW95] D. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50(3):551–573, June 1995.

[Lit89] N. Littlestone. From on-line to batch learning. In *Proc. 2nd Annu. Workshop on Comput. Learning Theory*, pages 269–284, San Mateo, CA, 1989. Morgan Kaufmann.

[LW94] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proc. 5th Annu. Workshop on Comput. Learning Theory*, pages 287–294. ACM Press, New York, NY, 1992.

[Val84] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[Vov90] V. Vovk. Aggregating strategies. In *Proc. 3rd Annu. Workshop on Comput. Learning Theory*, pages 371–383. Morgan Kaufmann, 1990.

APPENDICES

A Proof of Lemma 6

The Lemma is proved by induction on d. We first show that the statement is true for $d = 1$. In this case, upon expanding the value $v_n(2n, \ell+1, \ell)$ for the $\mathcal{G}(2n, \ell+1, \ell)$ game (see figure 1), we find

$$
\begin{aligned}
&v_n(2n, \ell+1, \ell) = \\
&\quad \inf_p \{pv(n, \ell, \ell-1) + (1-p)(\frac{1}{2} + v(2n, \ell, \ell))\} \\
&\quad = \inf_p \{1/2 + p(v(n, \ell, \ell-1) - 1/2)\}, \qquad (24)
\end{aligned}
$$

where (24) follows by noting that $v(2n,\ell,\ell)=0$. The thesis then follows from the fact that, by Lemma 4, $v(n,\ell,\ell-1)-\frac{1}{2}\leq 0$ and thus that $p=1$ minimizes (24).

Now, let $d-1\geq 0$ and assume that the thesis holds for the $\mathcal{G}(2n,\ell+d-1,\ell)$ game, i.e. for any $n\geq 1$ and for any $\ell\geq 1$, $v_n(2n,\ell+d-1,\ell)=v(n,\ell+d-2,\ell-1)$. We need to show that the assertion is also true for the $\mathcal{G}(2n,\ell+d,\ell)$ game. As with the case $d=1$, upon expanding the value $v_n(2n,\ell+d,\ell)$ we find

$$\begin{aligned} v_n(2n,\ell+d,\ell) &= \inf_p\{1/2+v(2n,\ell+d-1,\ell)+ \\ &\quad p[v(n,\ell+d-1,\ell-1)-1/2 \\ &\quad -v(2n,\ell+d-1,\ell)]\}. \end{aligned} \tag{25}$$

If we can show that the term multiplying p is non-positive then (25) is minimized when $p=1$. To show that this term is non-positive we use the following:

$$\begin{aligned} 1/2+v(2n,\ell+d-1,\ell) &\geq 1/2+v_n(2n,\ell+d-1,\ell) \\ &= 1/2+v(n,\ell+d-2,\ell-1) \\ &\geq v(n,\ell+d-1,\ell-1), \end{aligned}$$

concluding the proof. □

B Proof of Theorem 8

Throughout the proof we use the recursion presented in Corollary 7 for computing the value of the game. The Theorem is proved by using a double induction on d and ℓ. Since when only a single expert is left in the version space the value of the game is zero, throughout the proof we will assume $n\geq 2$. We first show that the statement is true when either $d=1$ or $\ell=0$ (base of the induction). The case $d=1$ is trivial. When $\ell=0$, i.e. the algorithm is left with 0 labels, the best strategy for the algorithm is to predict randomly thus $v(n,d,0)=\frac{d}{2}$ and $v(n,1,0)=\frac{1}{2}$ and the claim holds.

Now, let $d-1$ and $\ell-1$ be two arbitrary non negative integers and assume that for all $n\geq 2$,

1. $\forall \ell\geq 0,\ v(n,\ell+d-1,\ell)=(d-1)\,v(n,\ell+1,\ell)$,
2. $\forall d\geq 1,\ v(n,\ell-1+d,\ell-1)=d\,v(n,\ell,\ell-1)$.

We need to show that $v(n,\ell+d,\ell)=d\,v(n,\ell+1,\ell)$ is also true. We first prove that $v(n,\ell+d,\ell)\leq d\,v(n,\ell+1,\ell)$. Using the Remark of section 4.2 we can upper bound the value $v(n,\ell+d,\ell)$ of the $\mathcal{G}(n,\ell+d,\ell)$ game by

$$v(n,\ell+d,\ell)=\max_{\{i\leq\lfloor\frac{n}{2}\rfloor\}}\{v_i'(n,\ell+d,\ell)\}\leq$$

$$\max_{\{i\leq\lfloor\frac{n}{2}\rfloor\}}\{\frac{v(n-i,t-1,\ell-1)}{v(n-i,t-1,\ell-1)-v(i,t-1,\ell-1)+1}+$$

$$+\frac{(v(n-i,t-1,\ell-1)-v(i,t-1,\ell-1))v(n,t-1,\ell)}{v(n-i,t-1,\ell-1)-v(i,t-1,\ell-1)+1}\} \tag{26}$$

where in the above $t=\ell+d$. Noting that by the inductive hypothesis $v(n-i,\ell-1+d,\ell-1)=d\,v(n-i,\ell,\ell-1)$, $v(i,\ell-1+d,\ell-1)=d\,v(i,\ell,\ell-1)$ and $v(n,\ell+d-1,\ell)=(d-1)\,v(n,\ell+1,\ell)$, we can write equation (26) simply as

$$\begin{aligned} v(n,\ell+d,\ell) \leq \max_{\{i\leq\lfloor\frac{n}{2}\rfloor\}}\{&-\frac{d(d-1)\,v(i)\,v(n,\ell+1,\ell)}{d(v(n-i)-v(i))+1} \\ &+\frac{dv(n-i)(1+(d-1)v(n,\ell+1,\ell))}{d(v(n-i)-v(i))+1}\} \end{aligned} \tag{27}$$

where, for ease of notation, we have denoted $v(x,\ell-1,\ell)$ by $v(x)$. Substituting the expansion $v(n,\ell+1,\ell)=v(n-i^*)/(v(n-i^*)-v(i^*)+1)$ where $i^*=\arg\max_{\{i\in\mathbf{N}\,:\,i\leq\lfloor\frac{n}{2}\rfloor\}}\{v_i'(n,\ell+1,\ell)\}$ (see the Remark of section 4.2), into (27) we obtain

$$v(n,\ell+d,\ell)\leq\frac{d\,v(n-j)}{d(v(n-j)-v(j))+1}+$$

$$+\frac{d(d-1)\,v(n-i^*)(v(n-j)-v(j))}{(d(v(n-j)-v(j))+1)(v(n-i^*)-v(i^*)+1)}, \tag{28}$$

where $j\leq\lfloor\frac{n}{2}\rfloor$ maximizes the right hand side of (27). It is now sufficient to show that the right hand side of (28) is upper bounded by $d\,v(n,\ell+1,\ell)$. This is equivalent to prove that $f(i^*,j)\geq 0$ where

$$f(i^*,j)=\frac{d(v(n-i^*)(1-v(j))-v(n-j)(1-v(i^*)))}{(d(v(n-j)-v(j))+1)(v(n-i^*)-v(i^*)+1)}. \tag{29}$$

Since the denominator in (29) is always positive, it follows that $f(i^*,j)\geq 0$ if and only if $g(i^*,j)\geq 0$ where

$$\begin{aligned} g(i^*,j) &= v(n-i^*)(v(n-j)-v(j)+1) \\ &\quad -v(n-j)(v(n-i^*)-v(i^*)+1). \end{aligned} \tag{30}$$

Now, by definition of i^*, we have for any $j\leq\lfloor\frac{n}{2}\rfloor$

$$\frac{v(n-i^*)}{v(n-i^*)-v(i^*)+1}\geq\frac{v(n-j)}{v(n-j)-v(j)+1}$$

which implies $g(i^*,j)\geq 0$ and thus $v(n,\ell+d,\ell)\leq d\,v(n,\ell+1,\ell)$. Finally, we prove that $v(n,\ell+d,\ell)\geq d\,v(n,\ell+1,\ell)$ also holds by applying the following chain of inequalities

$$v(n,\ell+d,\ell)=\max_{\{i\leq\lfloor\frac{n}{2}\rfloor\}}\{v_i'(n,\ell+d,\ell)\}\geq v_j'(n,\ell+d,\ell)$$

$$\begin{aligned} &= \frac{v(n-j,\ell+d-1,\ell-1)\,v(n,\ell+d-1,\ell)}{v(n-j,\ell+d-1,\ell-1)-v(j,\ell+d-1,\ell-1)+1} \\ &\quad -\frac{v(j,\ell+d-1,\ell-1)\,v(n,\ell+d-1,\ell)}{v(n-j,\ell+d-1,\ell-1)-v(j,\ell+d-1,\ell-1)+1} \\ &\quad +\frac{v(n-j,\ell+d-1,\ell-1)}{v(n-j,\ell+d-1,\ell-1)-v(j,\ell+d-1,\ell-1)+1} \\ &= d\,v(n,\ell+1,\ell), \end{aligned}$$

where $j=\arg\max_{\{i\in\mathbf{N}\,:\,i\leq\lfloor\frac{n}{2}\rfloor\}}\{v_i'(n,\ell+1,\ell)\}$. This conclude the proof. □