TEACHING AND LEARNING UNDER DEDUCTIVE ERRORS

Anonymous authors
Paper under double-blind review

ABSTRACT

Most models of machine teaching and learning assume the learner makes no errors in its internal deductive inference. However, humans and large language models in few-shot learning regimes are two important examples of learners where this does not hold. They fail on some consistency checks, and they can fail stochastically. In this paper we introduce a teaching and learning framework that takes these deductive errors into account. We specifically study the case of machine teaching, as different characterizations of the teacher can account for both machine teaching and learning. In an overhauled Probably Approximately Correct (PAC) setting, we study theoretically that, for some estimated error level, the teacher must find a PAC teaching set that with high probability will lead the learner to guess a hypothesis that is approximately correct. We study the computational complexity of six different problems related to computing optimal PAC teaching sets. We give XP algorithms parametrized by size of teaching set, with tight runtime bounds under standard complexity assumptions like ETH. These results are complemented with a small experimental study of which teaching and learning protocols can best represent the observed behavior in some LLM teaching sessions.

1 INTRODUCTION

Learning mostly involves inductive inference, i.e., the process of going from facts to rules, from data to models. Many different paradigms for machine learning have been introduced according to how data is presented and how hypotheses are represented, and learning algorithms have been devised to make this process effective and efficient. However, most of these paradigms assume that the learner performs perfect deductive inference, i.e., the process of going from rules to facts, from models to data. In other words, learners are assumed to be consistent and complete when checking the alternative hypotheses during the learning process. Given a concept c in the hypothesis space C, and an example x, the learner invokes a consistency function $c(x)$ that returns true if the example belongs to concept c and false otherwise. We are not assuming that c is the correct hypothesis –that's the inductive inference search–, but we assume that if c were correct then the learner would be able to classify all examples perfectly with it. This assumption has seemed reasonable for learners that are implemented on computers with clear algorithms and representation languages whose deductive inference mechanisms are consistent and complete, e.g., applying a logistic regression to a new input, a neural network to a new image or a decision tree to a feature vector.

Following this tradition, machine *teaching*, where examples are chosen by the teacher, rather than randomly sampled from a distribution, has also assumed that the learner is deductively consistent and complete (Goldman and Kearns, 1995; Balbach, 2008; Zilles et al., 2011; Gao et al., 2017; Fallat et al., 2023). There are only very few examples in the literature where the representation is chosen in such a way that checking consistency becomes intractable or even incomputable (and hence the function cannot be complete) making approximations necessary (Ferri et al., 2026).

However, humans and pretrained large language models are two important examples of learners that should not be modelled with a perfect consistency function c. They fail on some consistency checks, and they can

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

fail stochastically. This is crucial for understanding few-shot learning and teaching with them, which is particularly relevant in explainable AI, as it is generally assumed that errors will originate from inductive inference but not deductive inference (Yang et al., 2021; Håvardstun et al., 2023). For these learners, we should consider they possess approximate consistency functions $\hat{c}_k(x_i)$ for each concept, whose error can be quantified by an error level γ_L as:

$$\gamma_L(c_k, x_i) = P[c_k(x_i) \neq \hat{c}_k(x_i)]$$

It is natural to consider that this error function depends on both the learner L, the example x_i and the concept c_k. For instance, it is easier to make a mistake when checking whether 8250715087503 is prime than whether it is even. On the other hand, it is also easier to make a mistake when checking whether 8250715087503 is prime than whether 27 is a prime.

Under this new perspective, traditional accounts of learning and teaching would simply not work. In particular, some of the simplest learning protocols, where the learner is given examples one after another and discards all the concepts that are inconsistent with the given evidence, until only one remains, would often fail catastrophically: a mistake in the consistency check at an earlier stage would rule out the right concept forever. Instead, we could keep all concepts and update the consistency vector for each concept as the process progresses, but would this lead to the identification of the right concept?[1]

We claim that the right approach for exploring the learnability and teachability in this paradigm is an adaptation of the Probably Approximately Correct (PAC) setting. Note that in classical PAC Learning (Valiant, 1984; 2013) the stochasticity comes from sampling of examples, whereas our focus is on the stochasticity arising from errors in consistency checking. Combining these two sources of stochasticity does not pose any fundamental problems, but it does complicate the presentation. Therefore, we will in this paper assume the setting of Machine Teaching (Zhu et al., 2018), where the examples given to the learner are chosen by a deterministic and consistent teacher rather than sampled, to highlight the new findings in this field that we will call PAC Teaching.

Because of the errors in consistency checking, the learner will never know for certain that the concept c being selected is the correct target concept c^*, not (only) because of the inductive inference problem, but because of a deductive inference fallibility, represented by the error level γ_L. It is then more natural to consider that the selected c can only be *approximately correct* even in cases of small hypothesis classes and a sufficient large number of examples. Also, because the error is of stochastic character, it also makes more sense to consider how *probable* it is that an approximately correct concept can be attained. The question becomes: given a learner L, characterized by a deductive error function γ_L and a particular learning protocol, can we devise teaching algorithms that lead to effective PAC teaching? In this paper we show that the answer is yes. After first defining the PAC teaching framework formally, and arguing for its suitability, we show in Section 5 that the problem of computing optimal PAC teaching sets, which comes in six different variants, can be solved by algorithms whose asymptotic runtimes are tight under standard complexity assumptions.

The rest of the paper is organized as follows.In Section 2 we define PAC Teaching formally. In Section 3 we argue that the PAC teaching framework is useful whenever the teaching success is inversely correlated with deductive errors, and show that this holds in experiments with LLMs as learners. In Section 4 we discuss various assumptions on learner and teacher behavior. We define a Prudent Learner which corresponds to an extension of the standard Empirical Risk Minimization learner to this new setting. In Section 5 we give algorithms and tight complexity bounds on the problems of computing optimal PAC teaching sets, and we also give some heuristic algorithms for computing PAC teaching sets.

[1]Note the learner can differ on $\hat{c}_k$ on different occasions, since γ_L is stochastic. We are not assuming that concepts are stochastic or there is some inherent noise in the world, but that checking of consistency of concepts is stochastic.

2 DEFINITIONS

As in classical Machine Teaching, we have a binary consistency matrix between an example set X and a concept set C, and denote by $c(x)$ the class label 1 or 0 (In or Out, True or False) of example $x \in X$ for concept $c \in C$. An example x labelled $b \in \{0, 1\}$, given as (x, b), is consistent with c if $c(x) = b$. We have a target concept, and the goal in classical machine teaching is to find a small set of labelled examples, the so-called teaching set S, such that the only concept that is consistent with all of S is the target concept.[2]

To account for learners that are not perfect when doing consistency checks we introduce mutually independent error probabilities $\gamma_L : C \times X \to [0, 1]$ where $\gamma_L(c, x)$ is the probability of learner L making an error when checking the value of $c(x)$. In the setting we introduce, when the learner L is given example x labelled b the error probability $\gamma_L(c, x)$ is used to decide if concept c is *L-consistent* with (x, b) or not. If indeed $c(x) = b$ then the learner concludes with probability $1 - \gamma_L(c, x)$ that c is L-consistent with this labelled example (and with probability $\gamma_L(c, x)$ that c is not L-consistent), while if $c(x) \neq b$ we have the opposite (the learner concludes with probability $\gamma_L(c, x)$ that c is L-consistent and probability $1 - \gamma_L(c, x)$ that c is not L-consistent). We assume throughout that, given the error matrix γ_L, all consistency checks are mutually independent Bernoulli trials, and checking c against (x, b) errs with probability $\gamma_L(c, x)$. This imperfect and stochastic version of c that L has will be denoted as $\hat{c}_L$ or simply $\hat{c}$. Note these errors can vary for each example and concept pair, but are symmetric with respect to checking for a correct or incorrect label, as the source of error is only tied to checking the value of $c(x)$.

For a concrete case, consider a concept class of subsets of natural numbers and teaching target concept $c^* = \texttt{ending-in-7}$ using the example $x = 27$ and checking consistency with $c = \texttt{primes}$, a pair for which $\gamma_L(c, x) = 0.1$. The learner is given $(x = 27, b = 1)$ and even though $c(x) = 0$ so c is not consistent there is a 10% chance that the learner will conclude that c is L-consistent, because of the error involved in $\hat{c}(x)$, in effect mistaking 27 as being prime, and if so reinforcing the belief that c could be a good concept for the unknown c^*, when it is not. A teacher aware of this could instead have chosen an example of lower error to rule out c, say $(x = 13, b = 0)$ with $\gamma_L(c, 13) = 0.01$.

Informally, in PAC teaching, when using a good teaching set S for the target concept c^*, it should be **probable** that the concept c chosen by the learner (e.g., the concept that is L-consistent with most labelled examples in S), is **approximately** equal to the target c^*. Note that we have two distinct cases, that we may call teaching for **identification** or teaching for **employment**. When the goal is identification then we want the chosen concept c, which is *named* by the learner and used by a perfect deductive user (e.g., the teacher), to be consistent with the target c^* on as many examples as possible, i.e. with high expected value of the event $c(x) = c^*(x)$ for some distribution over examples, as measured by the following *similarity* function

$$sim(c, c^*) = \mathbb{E}_{x \sim \mathcal{D}(X)}[P(c(x) = c^*(x))] \;\; (= 1/|X| \textstyle\sum_{x \in X} \mathbb{1}[c(x) = c^*(x)] \text{if uniform } \mathcal{D}, \text{ finite } X)$$

However, when the goal is employment by the learners themselves, then, based on the reasonable assumption that the same level of errors the learner L makes in consistency checking occur not only during teaching but also in employment, we instead define a similarity function sim_L as follows

$$sim_L(c, c^*) = \mathbb{E}_{x \sim \mathcal{D}(X)}[P(\hat{c}(x) = c^*(x))]$$

In other words, measuring the probability of c being L-consistent with x when labelled by $c^*(x)$. If we expand using the connection between $\hat{c}(x)$ and $\gamma_L(c, x)$, and assume that the distribution over examples is uniform and the example set is finite, then we get the following

$$sim_L(c, c^*) = \frac{1}{|X|} \sum_{x \in X} \begin{cases} 1 - \gamma_L(c, x) & \text{if } c(x) = c^*(x) \\ \gamma_L(c, x) & \text{if } c(x) \neq c^*(x) \end{cases}$$

[2] As noted above, we could also allow the teaching set to be randomly sampled, for a better alignment with formal models of Machine Learning, at the expense of introducing another source of stochasticity.

In PAC teaching for identification, for a given target concept c^*, probability p, and approximation ratio q, the goal is to find a small teaching set S such that the concept $L(S)$ chosen by the learner L when given S satisfies Equation 1 (in PAC teaching for employment replace sim by sim_L)

$$P[sim(L(S), c^*) \geq q] \geq p \tag{1}$$

For identification teaching with approximation ratio q we define the *good* subset of concepts to be $G(c^*, q) = \{c \in C : sim(c, c^*) \geq q\}$, and say that these are q-similar to c^* (for employment replace sim by sim_L to get $G_L(c^*, q)$ that are q_L-similar to c^*). The Probability of q-Approximately Correct Teaching for identification (or employment) when using teaching set S is then the probability that $L(S) \in G(c^*, q)$ (or $L(S) \in G_L(c^*, q)$). This depends on the behavior of the learner L which will be discussed in Section 4. However, it should be clear that higher errors γ_L will lead to larger teaching sets for fixed p and q, as illustrated in Figure 2 in the Appendix.

3 Applications of PAC Teaching

In this section we argue that (deductive) errors in consistency checking occur with many (inductive) learners, in particular with LLMs, and that the PAC teaching framework is useful whenever the success of teaching is correlated with the deductive errors.

As discussed in the introduction, the learning process involves at least two types of errors on part of the learner, that we may call the deductive error (inference involving consistency checks) and the inductive error (inference in searching for the right hypothesis). What we may call the total error rate of the learning process depends on both types of errors, and possibly other types, for example the error related to understanding correctly the learning task. The new PAC teaching framework assumes that the deductive errors form an important part of the total whose influence is significant. In many cases this is clearly the case. What about using LLMs as the learner? Recent research on LLMs show reliability fluctuations that deviate from what we would expect given human criteria of difficulty, see e.g. Zhou et al. (2024). Can deductive errors be derived from LLMs, and are they amenable to the PAC teaching framework? To answer these questions we perform the following experiments:

- Derive errors of the LLM on some fixed consistency checks, and call these deductive errors.
- Extract total success of the LLM as learner over varying teaching sets, on a task involving the same consistency checks, in scenarios believed to have both low and high total error
- For given teaching sets, do we see a negative correlation between total success and the deductive error, and is the variation in this correlation explainable by varying total error?

A positive answer to the final question would indicate that deductive errors can be derived from LLMs, and that the utility of the PAC teaching framework holds also for LLMs as learners.

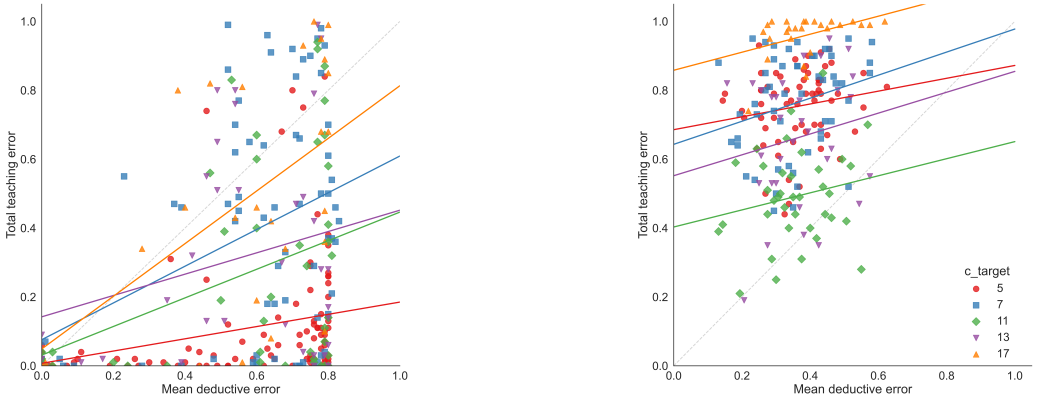
For experiments to answer these questions, we consider a concept class over positive integers.[3] Let $C = \{c_5, c_7, c_{11}, c_{13}, c_{17}\}$ and $X = \{1 \cdots 1000\}$, with $c_k(x) = 1$ iff x is a multiple of k. Note that for some prime numbers (2, 3, 5) there are easy algorithms deciding its multiples in base 10, and for this reason we included only one of them, namely 5, together with the next four prime numbers. We use GPT-5-nano (gpt-5-nano-2025-08-07) Singh et al. (2026) with reasoning effort set to minimal. Stronger models achieve near-perfect deductions on consistency for this concept class, leaving insufficient signal for illustrative purposes. GPT-5-nano falls in a capability category where it is strong enough to follow the format of inductive teaching but not so strong that checking for divisors becomes trivial. We use the default sampling parameters (temperature = 1, top_p = 1) to avoid introducing additional constraints on the model's output distribution.

[3] See Appendix for all System Prompts and see Github repository with all code: `https://anonymous.4open.science/r/Teaching_and_Learning_under_Deductive_Errors-B4A7/README.md`

We first describe how we measure the deductive error. In the selected concept class, consistency checking equates to checking 'Is k a divisor of x', for a given c_k and x. Hence, this is our user prompt. Each pair of $x \in X$ and k (for $c_k \in C$) is queried 20 times, and we calculate the deductive error $\gamma_L(c_k, x)$ as the fraction of times the LLM answered incorrectly (including nonsensical answers that form 0.3% overall[4]).

For the case of small total teaching error, we limit the experiment to teaching sets on a single positive example. In other words, define $X' = \{x \in X$: exactly one of 5,7,11,13,17 is a divisor of $x\}$ and use only teaching sets of type $\{x\}$ for $x \in X'$ such that the information uniquely identifies one concept. For each $x \in X'$, we query the LLM 100 times with the prompt 'Which of 5, 7, 11, 13 and 17 is a divisor of {x}?' and call the fraction of incorrect answers the *total teaching error* for this teaching set $\{x\}$.

From all this error data we get the plot in Figure 1 (a), where each point is a single x value, whose color and shape determines which $c_k \in C$ is the target (i.e. which unique k that divides x). The horizontal axis is the mean deductive error $1/5 \sum_{c_k \in C} \gamma_L(c_k, x)$ for x, and the vertical axis is the total teaching error for $\{x\}$.



(a) Low total error. Teaching set of size 1.

(b) High total error. Teaching set of size 2.

Figure 1: Each data point is a teaching set, in (a) of size one: a single positive x, and in (b) of size two: one negative x_1 and one positive x_2. Total teaching error on the vertical axis. On the horizontal axis we have mean deductive error (averaged over all 5 values of k) for x in (a) and averaged over x_1 and x_2 in (b). Each linear fit shows a positive correlation, which is required for the PAC teaching framework to be applicable, and this is more pronounced for low total error in (a), as expected.

Note the top left corner is empty, thus low deductive error predicts low total teaching error, and for each value of k the increasing linear fit shows that deductive errors influence total teaching error, in line with the PAC teaching framework. However, we also observe a cluster in the bottom-right corner, with high deductive error but low total teaching error. We attempt an explanation by example: checking if 730 is a multiple of 7 has a relatively high deductive error, but the teaching prompt specifies that only one of 7 and 5 is a divisor of 730 so the LLM may confirm that 5 is a divisor without even trying 7, and hence $\gamma_L(7, 730)$ does not impact the total teaching error. This hypothesis is supported by the fact that 5 has more points in this corner and also the lowest linear fit.

While low total teaching error can occur even when mean deductive error is high, the key finding is that selecting examples with low deductive error reliably yields low teaching error. This experiment, in a scenario of low inductive effort, thus demonstrates a concrete case in which a teacher using teaching sets of low

[4]Examples of nonsensical answers: Answer = None(...) , Answer = Benjamin(...), Answer = Delete this line(...) .

deductive errors can achieve a low total teaching error (often 0% and always below 10%) whereas a teacher oblivious to deductive error would risk a high total teaching error (often over 50% and sometimes 100%).

We now move to a scenario with the same deductive errors but with higher total teaching error due to higher inductive load. We use the same concept class C and example set X but with teaching sets of size two, thus coming in pairs with one positive and one negative, with both examples needed to uniquely identify the target. Thus, note that there is no element in X', used for teaching sets of size one, that will be used now. To construct these pairs, we first select all positive examples that are consistent with at least two concepts. For each such example, we then pair it with a negative example that is consistent with the target concept but inconsistent with the remaining competing concepts, thereby eliminating them and ensuring that only the target is identified. As before, for each such pair we repeat 100 times, now with the following prompt: 'Which of 5, 7, 11, 13 and 17 is not a divisor of x_1, while it is a divisor of x_2?', to extract total teaching error. See the plot in Figure 1(b) and compare it to 1(a). The results confirm our hypothesis: total error is higher, the outliers in the lower right quadrant have disappeared, and the correlation between total and deductive errors is still positive even though it has softened as the inductive error has become more prominent.

For other LLMs and for other teaching tasks, the derivation of deductive errors and total teaching error will have to change. The important finding is that these experiments support the hypothesis outlined at the start of this section, namely that deductive errors can be derived from LLMs as learners and that they are amenable to the PAC teaching framework.

4 Learner and teacher behavior

In this section we first specify two learner behaviors: a naive learner not aware of errors and a prudent PAC learner who is aware. Finally, three teacher behaviors are specified: a naive teacher not aware of errors, a heuristic PAC teacher and an optimal PAC teacher.

Firstly, we may have a small or large concept class, in an extreme case even infinite, and we may also have a prior distribution over the concepts. Somehow, the learner will consider a finite subset of concepts, that we may call $\tilde{C}$, possibly $\tilde{C} = C$ for a small concept class, or $\tilde{C}$ is achieved by sampling from C until it has a reasonable size containing a representative set of concepts including c^*. We will not go into details of how this is done, as it would vary based on the application. From now on, we simply assume $\tilde{C} = C$, and note that this may be altered depending on the application.

The **Naive learner** L_N acts unaware of any errors being made in consistency checking. This learner initializes the set of possible guessed concepts to $P = C$, goes through each labelled example $(x, b) \in S$, that is received as a batch, and discards any concept $c \in P$ that is not L-consistent, i.e. keeping c only if $\hat{c}(x) = b$ [5]. When done we define $L_N(S)$ to be an arbitrary concept in P, unless P is empty and L_N reports failure.

The **Prudent PAC learner** L_P is cognizant that errors can happen, so to avoid discarding the target it will for each concept $c \in C$ compute the count $Ct(c, S)$ of the number of labelled examples in S that c is L-consistent with, and arbitrarily guess a concept $L_P(S)$ with highest count. In other words, if the max count is $k = max_{c \in C} Ct(c, S)$ the prudent learner selects a concept $L_P(S)$ arbitrarily from $\{c \in C : Ct(c, S) = k\}$. We make no assumptions about how this selection is made, and for a worst-case scenario we assume that if there is some concept that is not good among those with highest count it would have been chosen.

Let us now consider the teacher. The teacher knows the errors γ_L made by the learner during consistency checking, and needs to find a teaching set S making it probable that the concept $L_P(S)$ guessed by the prudent learner is approximately equal to the target c^*, either by $L_P(S)$ being q-similar to the target for identification teaching, or $L_P(S)$ being q_L-similar to the target for employment teaching. As is standard in

[5] If $\hat{c} = c, \forall c \in C$ this is the standard Version Space learner in Machine Teaching.

Machine Teaching (and Machine Learning) the yardstick is the size of the teaching set, and for PAC Teaching this goal can be formulated in several ways. For fixed q and p the goal is to find a smallest teaching set S satisfying Equation 1, whereas fixing p and max-size k the goal is to find a teaching set S of size at most k that for the highest possible q satisfies Equation 1, and for fixed q and k the goal is to find a teaching set S of size at most k that for the highest possible p satisfies Equation 1.

Note that we may have a small or large example set, in an extreme case even infinite, and we may also have a distribution over the examples. Somehow, the teacher will consider a finite subset of examples to include in the teaching set, that we could call $\tilde{X}$, possibly $\tilde{X} = X$ for a small example set or $\tilde{X}$ is achieved by sampling from X until it has a reasonable size containing a representative set of examples. We will not go into details of how this is done. From now on, we simply assume $\tilde{X} = X$, and note that this may be altered depending on the application.

Let us start with the **Naive teacher** T_N who acts as if there were no errors (as if $\gamma_L(c, x) = 0$ always) and finds a teaching set S labelled according to c^* so that an error-free Version Space learner, who discards any concept not consistent, would be left only with c^*.

A **Heuristic PAC teacher** T_H will on the other hand use some heuristic based on the errors in γ_L and the behavior of L_P for picking examples to include in S. In Section 5 we discuss such heuristics.

Let us illustrate the benefits of PAC teaching with a very simple case highlighting the difference between the naive learner with a naive teacher versus the prudent learner with a heuristic teacher. Consider two concepts and two examples where $c_1(x_1) = c_2(x_2) = 1$, and $c_1(x_2) = c_2(x_1) = 0$ with concept-independent errors $\gamma_L(c_1, x_1) = \gamma_L(c_2, x_1) = 0.1$, and $\gamma_L(c_1, x_2) = \gamma_L(c_2, x_2) = 0.2$. For target c_2 the naive teacher T_N could use teaching set $S = \{(x_2, 1)\}$ and get a probability of 0.64 that $L_P(S) = c_2$, i.e. that c_2 is the only concept with max L-consistency count. A heuristic PAC teacher T_H, noticing that $\gamma_L(c_2, x_1) < \gamma_L(c_2, x_2)$ would instead use $S = \{(x_1, 0)\}$ for probability 0.81, and if requiring higher odds would use $S = \{(x_1, 0), (x_2, 1)\}$ for probability 0.8928.

Finally, we introduce the **Prudent-optimal PAC teacher** T_P aiming to find an optimal teaching set S for L_P, depending on what is the optimization criterion: given q and p minimize size k of S, or given q and k maximize the achievable p, or given p and k maximize the achievable q? Let us define this formally, both for identification (id) and for employment (em), for a total of six distinct optimization goals. Note each of these assumes the prudent learner L_P, but for ease of presentation we do not specify this in the notation used.

Firstly, let $0 \leq q, p \leq 1$ and k a positive integer and say that S is (q, p, k)-PAC-id if $|S| \leq k$ and $L_P(S)$ satisfies Equation 1. Likewise, say (q, p, k)-PAC-em when replacing sim by sim_L. Next, say that S is $(*, p, k)$-PAC-id optimal if it is (q', p, k)-PAC-id for some q' and no S' is (q'', p, k)-PAC-id for any $q'' > q'$. Also, say that S is $(q, *, k)$-PAC-id optimal if it is (q, p', k)-PAC-id for some p' and no S' is (q, p'', k)-PAC-id for any $p'' > p'$. Also, say that S is $(q, p, *)$-PAC-id optimal if it is $(q, p, |S|)$-PAC-id and no S' is (q, p, k)-PAC-id for any $k < |S|$. Analogously we define three PAC-em optimal sets. The optimal teacher T_P thus comes in six versions, three for identification as defined below, and three analogous ones (probable-em-, approx-em-, and size-em-) for employment.

- the probable-id-optimizer takes as input q, k and finds S that is $(q, *, k)$-PAC-id optimal
- the size-id-optimizer takes as input q, p and finds S that is $(q, p, *)$-PAC-id optimal
- the approx-id-optimizer takes as input p, k and finds S that is $(*, p, k)$-PAC-id optimal

5 Computing optimal and heuristic PAC Teaching sets

We start by investigating the computational complexity of the task faced by the prudent-optimal teacher T_P. We achieve tight runtime bounds for all six versions of this task: teaching for identification or employment

while optimizing for size or approximation or probability. We give algorithms whose runtimes provide upper bounds matching the lower bounds given by standard complexity assumptions.

The algorithms can be adapted for identification teaching or employment teaching simply by switching between similarity functions sim and sim_L. Firstly, note that in time $O(|C||X|)$, for a fixed approx-factor q and target c^*, we compute the set of Good concepts q-similar (or q_L-similar) to c^*. The following Lemma for a given teaching set S is very useful. Its proof is based on a dynamic programming algorithm that fills a size $|S|(|S|+1)$ DP table where $DP[i][j]$ holds the probability that a fixed concept is L-consistent j times on the $i+1$ first examples[6].

Lemma 1. *Given consistency matrix $C \times X$ and γ_L, teaching set S, target concept c^* and approx factor q, and Good concepts q-similar (or q_L-similar) to c^*, compute the probability that $L_P(S)$ is Good in time $O(|C| \cdot |S|^2)$.*

This lemma is used as a subroutine to give XP algorithms, parametrized by max-size k, for the probable-id-optimizer and the probable-em-optimizer problems. We also show that the runtime of these algorithms cannot be improved to the point of removing k from the exponent, unless the W-hierarchy collapses.

Theorem 1. *The problem facing the Prudent-optimal PAC Teacher T_P for probable-optimizing (id or em), given consistency matrix $C \times X$ and γ_L, approx factor q and max-size k, for a given target, can be solved in time $O(k^3|C||X|^k)$. Moreover, when parametrized by k this problem is $W[2]$-hard.*

By similar techniques, we are also able to give XP algorithms for the approx-id-optimizer and the approx-em-optimizer problems, parametrized by max-size k. Again, we show that the runtime cannot be improved to the point of removing k from the exponent, unless the W-hierarchy collapses.

Theorem 2. *The problem facing the Prudent-optimal PAC Teacher T_P for approx-optimizing (id or em) within a fixed precision (say d decimal digits), given consistency matrix $C \times X$ and γ_L, prob factor p and max-size k, for a given target, can be solved in time $O(k^3|C||X|^k)$. Moreover, when parametrized by k this problem is $W[2]$-hard.*

We also give algorithms for the size-id-optimizer and size-em-optimizer problems. These are again shown to have runtimes that are asymptotically optimal under standard complexity assumptions, in this case the Exponential Time Hypothesis and the Strong Exponential Time Hypothesis.

Theorem 3. *The problem facing the Prudent-optimal PAC Teacher T_P for size-optimizing (id or em), given consistency matrix $C \times X$ with $|X| = m$ and γ_L, approx factor q and prob factor p, for a given target, can be solved in time $O(m^2|C|2^m)$. Moreover, under ETH no algorithm with runtime $2^{o(m)}$ exists, and under Strong ETH no algorithm with runtime $(2-\epsilon)^m$ exists for any $\epsilon > 0$.*

We now turn to algorithms for T_H, i.e. heuristics computing good teaching sets for an error-aware learner like L_P. One option is to focus on the **Probably** Correct aspect of PAC Teaching. Intuitively, we want to teach with examples that *uniquely identify* the target, at least with few concepts sharing the label of the target on this example. For a target concept c^* first go over each example $x \in X$ and calculate, for some distribution over concepts, the expected value of the event that c is L-consistent with $(x, c^*(x))$. We call this the Uniqueness of x.

$$Uniqueness(x) = \mathbb{E}_{c \sim \mathcal{D}(C)}[P(\hat{c}(x) = c^*(x))]$$

Then sort the examples by the Uniqueness value and construct a teaching set greedily adding examples of lowest value first.

Another option is to focus on the **Approximately** Correct aspect of PAC Teaching. Intuitively, we want to teach with an example x' such that for the set of concepts $C_{x'}$ that share the label of the target on this example, it holds that these concepts are similar to the target. For a target concept c^* first go over each example

[6]Proofs of the lemma and all theorems can be found in the Appendix.

$x' \in X$ and calculate the set of concepts $C_{x'} = \{c \in C : c(x') = c^*(x')\}$, and then for some distribution over examples, calculate the expected value of the event that c drawn from $C_{x'}$, by some distribution, is L-consistent with $(x, c^*(x))$. We call this the Homogeneity-value of x'. Then sort the examples according to Homogeneity-value and construct a teaching set greedily adding examples of highest value first.

We implemented and ran these two heuristics on the concept class over multiples-of-k given in the plots of Figure 1. In the below table we see the results for the case of a teaching set of size 1. Both heuristics select a teaching set x of very low total teaching error of at most 0.02, for all 5 values of k, showing their usefulness in this case. When deductive error is less influential, see Table 2 for teaching sets of size two in the Appendix, these heuristics should be less useful.

Table 1: Selected teaching example 'x' (teaching size 1) by heuristic algorithm. The selected teaching examples have consistently low teaching error across all target concepts.

c_{target}	Criterion	x	Heuristic Score	Mean Deductive Error	Teaching Error
5	Uniqueness	10	0.0	0.00	0.02
	Homogeneity	10	0.124	0.00	0.02
7	Uniqueness	14	0.0	0.00	0.00
	Homogeneity	14	0.089	0.00	0.00
11	Uniqueness	11	0.0	0.00	0.02
	Homogeneity	305	0.086	0.79	0.00
13	Uniqueness	13	0.0	0.00	0.00
	Homogeneity	132	0.073	0.76	0.02
17	Uniqueness	34	0.0	0.00	0.00
	Homogeneity	34	0.077	0.00	0.00

We can also combine to give a heuristic for the **Probably and Approximately** Correct aspect of PAC Teaching. Calculate a combined value for each example, based on some weighting between having low Uniqueness and simultaneously having high Homogeneity, say $Combined(x) = (1 - Uniqueness(x)) + \alpha \times Homogeneity(x)$ for a chosen factor α. Then sort by these values and construct a teaching set greedily adding examples of highest $Combined$ value first.

6 Conclusion

We have introduced the PAC teaching framework, that is useful whenever we have a learner who does not perform perfect deductive inference. We have defined different behaviors of both the teacher and the learner in situations where these errors are stochastic and measurable. We have given algorithms finding the optimal teaching sets in these situations, and shown that the runtime of these algorithms cannot be improved unless standard complexity assumptions fail.

The PAC teaching framework is useful even if we do not know the algorithms used by the learner. This is clear from Table 1, where the LLM is not applying a learning algorithm known to us, but after deriving its deductive errors we are able to use a heuristic algorithm to compute teaching sets that increase the chance of successful PAC teaching.

AI use statement

No so-called generative AI was used at any stage of this paper, neither in research nor writing nor any other part, except for the experiments with LLMs described in Section 3.

REFERENCES

F. J. Balbach. Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science*, 397(1-3):94–113, 2008.

C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *International Workshop on Parameterized and Exact Computation*, pages 75–85. Springer, 2009.

M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi: 10.1145/2925416. URL https://doi.org/10.1145/2925416.

R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

S. Fallat, D. Kirkpatrick, H. U. Simon, A. Soltani, and S. Zilles. On batch teaching without collusion. *Journal of Machine Learning Research*, 24:1–33, 2023.

C. Ferri, D. Garigliotti, J. Hernandez-Orallo, B. Håvardstun, and J. A. Telle. When redundancy matters: Machine teaching of representations. *Machine Learning*, 115(1):20, 2026.

Z. Gao, C. Ries, H. U. Simon, and S. Zilles. Preference-based teaching. *Journal of Machine Learning Research*, 18:31:1–31:32, 2017. URL http://jmlr.org/papers/v18/16-460.html.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

B. A. T. Håvardstun, C. Ferri, J. Hernández-Orallo, P. Parviainen, and J. A. Telle. XAI with machine teaching when humans are (not) informed about the irrelevant features. In *ECML/PKDD*, 2023. URL https://api.semanticscholar.org/CorpusID:262089041.

A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–348, 1991.

A. Singh, A. Fry, A. Perelman, et al. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

L. Valiant. *Probably approximately correct: nature's algorithms for learning and prospering in a complex world*. Basic Books, 2013.

L. G. Valiant. A theory of the learnable. *Comms. of the ACM*, 27(11):1134–1142, 1984.

S. C.-H. Yang, W. K. Vong, R. B. Sojitra, T. Folke, and P. Shafto. Mitigating belief projection in explainable artificial intelligence via bayesian teaching. *Scientific reports*, 11(1):1–17, 2021.

L. Zhou, W. Schellaert, F. Martínez-Plumed, et al. Larger and more instructable language models become less reliable. *Nature*, 634:61–68, 2024. doi: 10.1038/s41586-024-07930-y.

X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

S. Zilles, S. Lange, R. Holte, and M. Zinkevich. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12(Feb):349–384, 2011.

7 APPENDIX

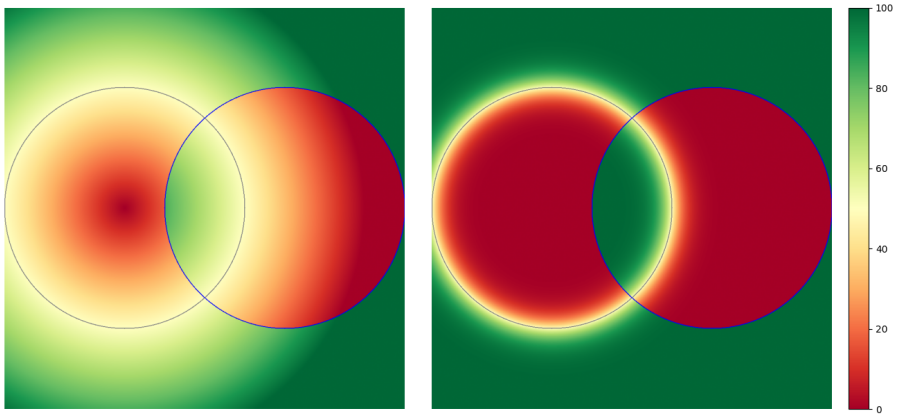


Figure 2: For fixed p, q higher errors give larger PAC teaching sets. In both figures above the concept class consists of circles in the plane with $c(x) = 1$ when point x is within circle c, with only two such concepts shown: target c^* the circle on the right, and c' the circle on the left. The color of a point x gives the probability of c' being L-consistent with x when labelled by $c^*(x)$, from high probability in dark green to low probability in dark red. Thus, the more green and less red the higher the L-similarity between c' and the target. For the figure on the right we have non-zero error $\gamma_L(c', x) > 0$ only when x is very close to perimeter of c'. The symmetric difference is then dark red, apart from the perimeter of c' where the error kicks in. In the left figure $\gamma_L(c', x)$ is higher, and proportional to the distance from x to perimeter of c', and the teacher has less examples to choose from to rule out c' with high probability. If many circles had this same error then more teaching examples would be needed to arrive with high probability at a circle that closely approximates the target (small symmetric difference), as any single example would be dark red for only few circles.

7.1 LLM PROMPTS

Here are the prompts used in our LLM experiments. See also the Github repository mentioned in the footnote.

Deductive Error - System prompt

```
Answer format:  'Answer = Yes' or 'Answer = No'
```

Deduclearnertive Error - User message

```
Is {k} a divisor of {x}?
```

Low Inductive Load - Teaching System prompt

```
Answer format:  Answer = 5 or Answer = 7 or Answer = 11 or Answer = 13 or
Answer = 17
```

learner

Low Inductive Load - Teaching User message

```
Which of 5, 7, 11, 13 and 17 is a divisor of {x}?
```

High Inductive load - Teaching System prompt

```
Answer format:  Answer = 5 or Answer = 7 or Answer = 11 or Answer = 13 or
Answer = 17
```

High Inductive load - Teaching User message

```
Which of 5, 7, 11, 13 and 17 is not a divisor of {x1}, while it is a divisor
of {x2}?
```

7.2 Proofs of Lemma and Theorems

Here is the proof of Lemma 1.

Proof. In Algorithm 1 we give the pseudo code. Note that there are five outer for loops.

Let us first argue for correctness. We apply a top-down approach, arguing from the fifth and last outer for loop to the first one, and hence start with restating the goal. With G the set of Good concepts and $B = C \setminus G$, we want to know the probability that a concept in G is L-consistent more times than all concepts in B. We go over all the possible scenarios where at least one concept in G is L-consistent at least once more than any concept in B, computed by the following formula

$$P[max_{g \in G} X_g > max_{b \in B} X_b | S] = \\ = \sum_{i=1}^{|S|} P[max_{g \in G} X_g = i | S] \cdot P[max_{b \in B} X_b \leq i - 1 | S]$$

In the above X_c is simply shorthand for the count $Ct(c, S)$, and the reader can check that this formula corresponds to the fifth and last outer for loop of the pseudo-code.

For sake of simplicity we omit the repeated S from the rest of the equations. The fourth outer for loop computes the probability that the highest count over all concepts in C equals i, and the formula for this is given below. In the below equation we use a general set of concepts S, which in the pseudo-code is switched out with G or B depending on the calculation.

$$P[max_{c \in S} X_c = i] = P[max_{c \in S} X_c \leq i] - P[max_{c \in S} X_c \leq i - 1]$$

with the base case

$$P[max_{c \in S} X_c = 0] = P[max_{c \in S} X_c \leq 0]$$

The third outer for loop computes the probability that the highest count is *at most* i and the formula is given below

Algorithm 1 Calculate probability of Approximately Correct teaching for a given teaching set. Runtime $O(|C| \cdot |S|^2)$

Input: S, C, γ_L, c^* and q, yielding a partition of concepts into G good and B bad that we also assume as input.

$k \leftarrow |S|$
for $c \in C$ **do** ▷ $O(|C| \cdot |S|^2)$
  $PMF_c = DP_{sub}(c, S)$ ▷ $O(|S|^2)$
  for $i \in \{0 \cdots k\}$ **do** ▷ $O(|S|)$
    $P[X_c = i] = PMF_c[i]$
  end for
end for
for $c \in C$ **do** ▷ $O(|C| \cdot |S|)$
  $P[X_c \leq 0] = P[X_c = 0]$
  for $i \in \{1 \cdots k\}$ **do** ▷ $O(|S|)$
    $P[X_c \leq i] = P[X_c = i] + P[X_c \leq i - 1]$
  end for
end for
for $i \in \{\ 0 \cdots k\}$ **do** ▷ $O(|G| \cdot |S| + |B| \cdot |S|) = O(|C| \cdot |S|)$
  $P[max_{g \in G} X_g \leq i] = 1$
  $P[max_{b \in B} X_b \leq i] = 1$
  for $g \in G$ **do** ▷ $O(|G|)$
    $P[max_{g \in G} X_g \leq i] * = P[X_g \leq i]$
  end for
  for $b \in B$ **do** ▷ $O(|B|)$
    $P[max_{b \in B} X_b \leq i] * = P[X_b \leq i]$
  end for
end for
$P[max_{g \in G} X_g = 0] = P[max_{g \in G} X_g \leq 0]$
for $i \in \{1 \cdots k\}$ **do** ▷ $O(|S|)$
  $P[max_{g \in G} X_g = i] = P[max_{g \in G} X_g \leq i] - P[max_{g \in G} X_g \leq i - 1]$
end for
$Probability = 0$
for $i \in \{1 \cdots k\}$ **do** ▷ $O(|S|)$
  $Probability + = P[max_{g \in G} X_g = i] \cdot P[max_{b \in B} X_b \leq i - 1]$
end for
return $Probability$

Algorithm 2 $DP_{sub}(c, S)$. Find $DP[i][j]$, probability that c is L-consistent j times on the $i+1$ first examples of S, for some $0 \leq i \leq |S|-1, 0 \leq j \leq |S|$.

Input: c, S, γ_L
  $keep(x, c) = (1 - \gamma_L(c, x))$ if label of x is same as $c(x)$
  $keep(x, c) = \gamma_L(c, x)$ if label of x is different from $c(x)$
  $k \leftarrow |S|$
  $DP \leftarrow [k][k+1]$ ▷ Initialization
  $DP[0][0] \leftarrow 1 - keep(S[0], c)$
  $DP[0][1] \leftarrow keep(S[0], c)$
  for $j \in \{2 \cdots k\}$ **do**
    $DP[0][j] \leftarrow 0$
  end for
  for $i \in \{1 \cdots k-1\}$ **do**
    $DP[i][0] \leftarrow DP[i-1][0] \cdot (1 - keep(S[i], c))$
  end for
  for $i \in \{1 \cdots k-1\}$ **do**
    for $j \in \{1 \cdots k\}$ **do**
      $DP[i][j] \leftarrow DP[i-1][j-1] \cdot keep(S[i], c) + DP[i-1][j] \cdot (1 - keep(S[i], c))$
    end for
  end for
  for $j \in \{0 \cdots k\}$ **do**
    $P[X_c = j] \leftarrow DP[k-1][j]$
  end for

$$P[max_{c \in S} X_c \leq i] = \prod_{c \in S} P[X_c \leq i]$$

The second outer for loop calculates the probability of a concept c having count at most i, and for this we use the Cumulative Distribution Function CDF

$$P[X_c \leq i] = P[X_c \leq i-1] + P[X_c = i]$$

with the base case

$$P[X_c \leq 0] = P[X_c = 0]$$

Finally the first outer for loop calculates the probability that a single concept c has count exactly i, namely

$$P[X_c = i] \quad \forall_{i \in \{0 \cdots |S|\}}$$

In our pseudo-code this is done by a straightforward Dynamic programming called $DP_{sub}(c, S)$ given as a separate Algorithm 2. This DP fills a $|S|(|S|+1)$ size table where $DP[i][j]$, for some $0 \leq i \leq |S|-1, 0 \leq j \leq |S|$, holds the probability that c is L-consistent j times on the $i+1$ first examples.

As we have argued along the way, these formulas compute what we need and they correspond to the pseudo-code. We are therefore done with the correctness proof.

We now argue for the runtime. In the first outer for loop, the Dynamic Programming called $DP_{sub}(c, S)$ has runtime $O(|S|^2)$, and since this is computed for each concept the first outer loop is $O(|C||S|^2)$. The second and third outer for loops are $O(|C||S|)$, while the fourth and fifth are $O(|S|)$. We conclude that the runtime is $O(|C||S|^2)$ □

Here is the proof of Theorem 1.

Proof. For given q, k we search for the best probable-optimizing teaching set by going over all subsets of size at most k, and for each one run the algorithm of Lemma 1 to find the largest value p such that this set is (q, p, k)-PAC-id (alt. (q, p, k)-PAC-em). The set S associated with the highest value p achieved this way is a $(q, *, k)$-PAC-id (alt. $(q, *, k)$-PAC-em) optimal set. The total runtime is $O(|C| \sum_{i=1}^{k} \binom{|X|}{i} \cdot i^2)$ which can be simplified to $O(k^3|C||X|^k)$.

Improving this runtime to an FPT algorithm, i.e. to remove k from the exponent, would imply a collapse of the W-hierarchy in Parameterized Complexity, as we now show. By a reduction similar to one given in Shinohara and Miyano (1991) we show that the probable-optimizing problem is W-hard. In the k-Hitting Set problem we are given an integer k and a family F of subsets of a universe U and asked if there are k elements of the universe that hit every subset in F, and this problem is W[2]-hard Downey and Fellows (2013). The parameterized reduction from Hitting Set to the probable-optimizing problem constructs an instance with example set $X = U$, and for each $A \in F$ a concept $c_A \in C$ such that for $x \in X$ we set $c_A(x) = 1$ if and only if $x \in A$. We add a target concept c^* with $c^*(x) = 0$ for all x, set $\gamma_L(c, x) = 0$ for all x, c, set $q = 1$, and crucially (for the parameterized aspect) use the value of k as the size bound. Assuming there was an FPT algorithm for the probable-optimizer parameterized by k, then given an instance of k-Hitting Set we follow the above construction and run this algorithm on the constructed instance and check if the optimal set returned was $(q, 1, k)$-PAC-id (or $(q, 1, k)$-PAC-em) optimal, i.e. with $L(S)$ guessing a 1-similar concept with probability 1, in effect guessing the target always. If that is the case we have a yes-instance of the Hitting set problem, and otherwise we have a no-instance. This would constitute an FPT algorithm for k-Hitting Set and concludes the proof of the Theorem. □

Here is the proof of Theorem 2.

Proof. For given p, k we search for the best approx-optimizing teaching set by using an outer loop that performs a binary search for the best value q within d decimal digits. This amounts to at most $log_2(10^d)$ iterations of the outer loop, which is $O(1)$ for constant d. For each such value of q we go over subsets of size at most k, and run the algorithm of Lemma 1, stopping as soon as the output probability p' is at least p, meaning this set is (q, p, k)-PAC-id (alt. (q, p, k)-PAC-em). When stopping this way we continue the binary search in the larger range, whereas if no set of size at most k gives the successful stop then we continue in the lower range. For the highest successful value of q the associated set S that forced the stop is a $(*, p, k)$-PAC-id (alt. $(*, p, k)$-PAC-em) optimal set. The total runtime is $O(|C| \sum_{i=1}^{k} \binom{|X|}{i} \cdot i^2)$ which can be simplified to $O(k^3|C||X|^k)$.

Improving this runtime to an FPT algorithm, i.e. to remove k from the exponent, would imply a collapse of the W-hierarchy in Parameterized Complexity, by a similar reduction as the one given above in the proof of Theorem 1. The difference to that reduction is only that now we set $p = 1$ and do the reduction to the case of fixed precision on a single decimal digit. Then, we have a yes-instance of k-Hitting Set if and only if the highest successful value returned is $q = 1.0$. Assuming there was an FPT algorithm for the approx-optimizer parameterized by k, there would also be one for k-Hitting Set. This concludes the proof of the Theorem. □

Here is the proof of Theorem 3.

Proof. For given q, p we search for the size-optimizing teaching set by going over all subsets and for each such S run the algorithm of Lemma 1 to check if S is $(q, p, |S|)$-PAC-id (alt. $(q, p, |S|)$-PAC-em). The smallest value k with a set S with $|S| = k$ giving a positive answer means a $(q, p, *)$-PAC-id (or $(q, p, *)$-PAC-em) optimal set of size k. The total runtime is $O(m^2|C|2^m)$.

It is known that the Hitting Set problem on a universe of size m cannot be solved in time $2^{o(m)}$ unless ETH fails Calabro et al. (2009), and under Strong ETH no algorithm with runtime $(2-\epsilon)^m$ for any $\epsilon > 0$ Cygan et al. (2016). We reduce Hitting Set to size-optimising teacher by giving a similar reduction as the one given above in the proof of Theorem 1, to show the lower bound stated in the Theorem. The difference to that reduction is only that now we fix $q = p = 1$ and the size-constraint varies with the size of S. Moreover, the answer is checked against the size of the smallest S returned and the Hitting Set is a yes-instance if and only if $|S| \leq k$, for the value of k given as part of the Hitting Set instance. Assuming there was an algorithm for the size-optimizer with runtime $O(2^{o(|X|)})$ then this would translate, through this reduction, to an algorithm solving Hitting Set in time $2^{o(m)}$, refuting ETH. Likewise, an algorithm for the size-optimizer with runtime $O^*((2-\epsilon)^{|X|})$ for $\epsilon > 0$ would translate, through this reduction, to an algorithm solving Hitting Set in time $O^*((2-\epsilon)^m)$, refuting Strong ETH. This concludes the proof of the Theorem. □

7.3 Table of results for the Heuristic algorithm

Table 2: Selected teaching set 'x_1, x_2' by heuristic algorithm in the scenario with high total teaching error. As deductive errors are less correlated with teaching error in this setting, the non-optimal teaching error resulting from these PAC teaching sets is expected.

c_{target}	Criterion	x_1, x_2	Heuristic Score	Mean Deductive Error	Teaching Error
5	Uniqueness	{35, 119}	0.13	0.23	0.77
	Homogeneity	(35, 119)	0.13	0.23	0.77
7	Uniqueness	{35, 660}	0.2	0.30	0.64
	Homogeneity	{455, 325}	0.2	0.43	0.94
11	Uniqueness	{55, 190}	0.3	0.18	0.39
	Homogeneity	{55, 190}	0.3	0.18	0.39
13	Uniqueness	{429, 374}	0.38	0.44	0.78
	Homogeneity	{845, 750}	0.38	0.38	0.65
17	Uniqueness	{357, 672}	0.28	0.50	1.00
	Homogeneity	{595, 385}	0.28	0.48	0.98