Artificial Intelligence 217 (2014) 198–215

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Eliciting good teaching from humans for machine learners

Maya Cakmak [a,*], Andrea L. Thomaz [b]

[a] *Computer Science and Engineering Department, University of Washington, 185 Stevens Way, Seattle, WA 98195, United States*
[b] *School of Interactive Computing, Georgia Institute of Technology, 801 Atlantic Dr., Atlanta, GA 30332, United States*

A R T I C L E   I N F O

Article history:
Received 12 January 2012
Received in revised form 18 August 2014
Accepted 24 August 2014
Available online 10 September 2014

Keywords:
Algorithmic teaching
Interactive machine learning

A B S T R A C T

We propose using computational teaching algorithms to improve human teaching for machine learners. We investigate example sequences produced naturally by human teachers and find that humans often do not spontaneously generate optimal teaching sequences for arbitrary machine learners. To elicit better teaching, we propose giving humans *teaching guidance*, which are instructions on how to teach, derived from computational teaching algorithms or heuristics. We present experimental results demonstrating that teaching guidance substantially improves human teaching in three different problem domains. This provides promising evidence that human intelligence and flexibility can be leveraged to achieve better sample efficiency when input data to a learning system comes from a human teacher.

1. Introduction

We are interested in the interactive Machine Learning (ML) scenario of a human training an agent to perform a classification task by showing examples. A diverse set of ML applications take input data directly from a human who is not a ML expert (*e.g.* document classification, user preference modeling, robot programming by demonstration). In these applications it is essential that ML algorithms require a minimal amount of data, as providing it can be a cumbersome process for the human. To this end, ML research has produced a range of methods that adapt conventional learners to improve their sample efficiency in learning (*e.g.* Active Learning or Semi-supervised Learning). The main idea that we explore in this article is to improve the *teacher*, rather than the learner, in trying to achieve this objective.

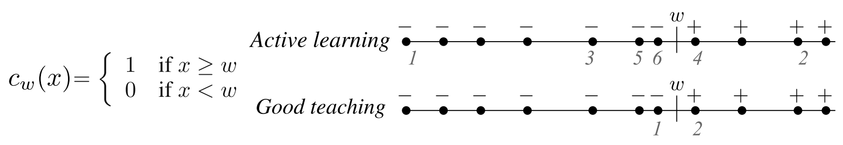
A helpful teacher can significantly improve the learning rate of a ML algorithm, as shown in the field of Algorithmic Teaching [3,29,15]. This field studies the *teaching problem*, that is, producing a set of labeled examples based on a *known* target concept. The goal is to find efficient algorithms that can teach with as few examples as possible. Oftentimes this is much smaller than the number of randomly, or even actively, chosen examples needed to learn a concept [2,15]. In other cases, a concept that is not PAC learnable under arbitrary distributions, may become learnable with a helpful teacher [10].

To illustrate the potential of good teaching, consider a canonical example from Dasgupta [9] (illustrated in Fig. 1). The learning problem is to find a threshold that separates two classes in 1-D space, from observed examples. A simple consistent learner achieves this by placing the threshold between the rightmost negative example and the leftmost positive example. In this setting an active learner[1] achieves logarithmic advantage over random sampling, by always querying the unlabeled

* Corresponding author.
E-mail addresses: mcakmak@cs.washington.edu (M. Cakmak), athomaz@cc.gatech.edu (A.L. Thomaz).

[1] Active Learning is a paradigm in which the learner chooses the instance that will be labeled by the teacher. Label requests made by the learner are called *queries*.

http://dx.doi.org/10.1016/j.artint.2014.08.005
0004-3702/© 2014 Elsevier B.V. All rights reserved.



$c_w(x) = \begin{cases} 1 & \text{if } x \geq w \\ 0 & \text{if } x < w \end{cases}$
Active learning
Good teaching
w
1
3
5
6
4
2
1
2

Fig. 1. Motivating example from [9]. The numbers indicate the order in which the examples are provided to the learner.

sample closest to the estimated boundary. However, a good teacher could directly provide the two examples closest to the true decision boundary, to achieve the smallest possible error rate.

Given the significant potential of good teaching we pose two questions:

- How well do humans naturally teach machine learners?
- Can we influence humans to teach more optimally?

In particular we are interested in non-expert humans who might not fully understand the inner workings of the learner they are teaching. Human teaching is largely optimized for human learning, and therefore is not expected to be naturally optimal for arbitrary machine learners. However, we believe that human teachers have the capacity to adapt for the needs of a specific learner. We propose to achieve this with *teaching guidance*.

Teaching guidance is intended to directly guide the teacher through instructions on how to teach. We propose grounding such instructions in computational solutions to the teaching problem at hand. In this article, we present a series of experiments that explore the use of computational solutions to different teaching problems, for guiding human teachers. Our experiments demonstrate that humans are not spontaneously as good as computational teachers that are tailored for the specific learner, and that teaching guidance improves their example sequences across all of our problem domains.

2. Teaching guidance

Teaching guidance is a set of instructions given to human teachers, with the goal of influencing their choice of examples towards those that are most informative for a particular learner. Teaching guidance is *specific* to a learner; however, it is *independent* of the particular concept that is being taught to the learner. The objective of teaching guidance is to improve upon natural teaching by human teachers.

The approach proposed in this article is to construct teaching instructions based on computational solutions to the teaching problem at hand. These solutions can be in the form of either an *algorithm* or a *heuristic*. Algorithms can have guaranteed optimality bounds; however, they are often not as amenable to be used as teaching guidance. Although heuristics may not guarantee optimality, they are often easier to understand and use for everyday people. In the rest of this section we describe these two types of computational solutions to the teaching problem and discuss how they can be translated into teaching guidance.

2.1. Algorithmic teaching guidance

The first type of teaching guidance involves explaining an *optimal teaching algorithm* to a human teacher. A teaching algorithm produces a sequence of labeled examples τ_c consistent with a *known* target concept c from the concept class $\mathcal{C}$. Formally, a *concept* is a function that assigns a label to all points in the state space, *i.e.* a separation of the state space. A *concept class* is a set of concepts, often with a compact representation.

The objective of the teacher is to allow the learner to uniquely identify the correct target concept. A concept is uniquely identified when it is the only concept in $\mathcal{C}$ that is consistent with a provided data set. The *optimal teaching* problem is to achieve unique identification with as few examples as possible. We refer to a shortest sequence of examples that uniquely identifies a target concept as an *optimal teaching sequence* for that concept. Note that more than one such sequences might exist. We denote the length of an optimal teaching sequence as $|\tau_c^*| = \min_{\tau \in \mathcal{T}_c} |\tau|$ where $\mathcal{T}_c$ is the set of example sequences that uniquely identify c. An algorithm that produces a teaching sequence of length $|\tau_c^*|$ for every concept $c \in \mathcal{C}$ is referred to as an *optimal teaching algorithm* for the concept class $\mathcal{C}$.

Characterizing the teachability of concepts is a central problem in Algorithmic Teaching, and many teachability metrics have been proposed. The most popular metric is the *Teaching Dimension* [15], which quantifies teachability of a concept class by the number of examples required to optimally teach the hardest-to-teach concept in the class:

$$TD(\mathcal{C}) = \max_{c \in \mathcal{C}}(|\tau_c^*|)$$

The teaching dimension of a particular concept c with respect to the concept class $\mathcal{C}$ is $TD(c, \mathcal{C}) = |\tau_c^*|$.

Finding an optimal teaching sequence for an arbitrary concept is NP-hard, by reduction to the *minimum set cover* problem [15]. However, polynomial time algorithms have been proposed for certain concept classes such as conjunctions, monotone decision lists, orthogonal rectangles and monotone K-term DNFs, among others. These algorithms cleverly ex-

Algorithm 1 Optimal teaching algorithm pseudo-code for conjunctions.

```
Require: Target concept $c_r \in \mathcal{C}^n$
  Show one positive example from $c_r$.
  if $r < n$ then
    Show another positive example from $c_r$ in which all $(n-r)$ irrelevant features are reversed.
  end if
  for all $r$ do
    Show a negative example in which one of the $r$ relevant features is reversed.
  end for
```

ploit the compact (polynomial-size) representation of concepts and instances, to avoid enumerating all possible example sequences.

For a large set of practical learning problems, there is no known optimal teaching algorithm. For these problems one can devise *approximately optimal* algorithms, which are sub-optimal algorithms whose solutions differ from optimal ones within provable bounds. One of the well-known approximate algorithms for the *set cover* problem is one that greedily selects the set with highest cover at every step. For a teaching algorithm, this can be interpreted as greedily choosing an example that reduces the version space the most, *i.e.* that rules out the most concepts. In problems where the version space cannot be enumerated, a greedy teaching algorithm could select the example that increases the learner's performance (*e.g.* accuracy) the most.

Although both optimal and approximately-optimal teaching algorithms could potentially serve as teaching guidance, generally, approximately-optimal algorithms are not feasible. The greedy algorithm described above requires the ability to simulate and evaluate the learner, as well as, the ability to sort all possible examples in terms of how much they would improve the learner when presented. As will be discussed in Section 2.2, heuristic algorithms are more suitable for teaching guidance in problems where no optimal teaching algorithms are known.

The operation of an optimal teaching algorithm is often intuitive to an educated viewer; however, getting everyday people to use the algorithm when they are providing examples to a machine can still be challenging. In the following, we present the optimal teaching algorithm for conjunctions and discuss how it can be used as teaching guidance.

2.1.1. *Teaching conjunctions*

A conjunction is a list of feature values that must all be true, for a sample to be labeled as positive. Features in the conjunction are referred to as *relevant*, while all the other dimensions of the sample space are *irrelevant* features. We focus on the general set of conjunctions with binary features, which can have either a feature or its negation in the conjunction (rather than *monotone* conjunctions which do not allow negations).

Consider the concept class $\mathcal{C}^n$ of conjunctions over n binary variables. The teaching dimension of a concept c_r with respect to this concept class is $TD(c_r, \mathcal{C}^n) = \min(n+1, r+2)$, where r is the number of relevant variables for c_r [15]. The algorithm that produces an optimal teaching sequence of this size is outlined in Algorithm 1. The first two positives examples chosen by the first two steps of the algorithm prove to the learner that all changed features are irrelevant.[2] The negative examples chosen by the loop in the algorithm show the learner that the feature changed in each iteration is relevant, since the negative example differs from a known positive example by only one feature.

In our experiments, the teaching guidance provided to humans for teaching a conjunction is a step-by-step strategy based on this algorithm. Some terms such as *positive/negative example* or *relevant/irrelevant features* need to be replaced or explained in such guidance. A domain specific example of guidance for teaching conjunctions will be seen in Experiment 1 (Section 3.3).

2.2. *Heuristic teaching guidance*

The second type of teaching guidance involves explaining a *teaching heuristic* to a human teacher. Many practical learning problems do not have known efficient optimal teaching algorithms, and the approximately optimal alternatives are intractable for human teachers. For these problems we propose using teaching heuristics that approximate the informativeness of an example for the learner. These heuristics aim to capture the intuition of an optimal teacher. A computational teacher can use this heuristic to rank all possible examples and greedily present the best example at each step. Although there are no optimality guarantees for these teachers, we expect they would be better than a random teacher. We also expect that, when the heuristic is described to human teachers as a way for selecting examples, it will lead to better teaching examples than those humans give naturally.

One of the main assumptions for computational heuristic teachers is that the set of all possible examples, $\mathcal{S}_c$, is finite. The teaching heuristic is therefore a means of *ordering* or *ranking* these examples in terms of their informativeness. We first consider teaching heuristics that can be used as teaching guidance in such problems where the teacher selects examples from a finite set. Later, we extend the idea of heuristic teaching guidance to problems that involve generating samples, rather than selecting them from a pool.

[2] Note that if $r = n$ then all features are relevant and there is only one positive example, thus $|\tau^*_{c_r}| = n + 1$.

Algorithm 2 Pseudo-code for the heuristic teaching algorithm.

```
Require: Set of examples $\mathcal{S}^c = \{(x_i, y_i)\}_{i=1}^N$ labeled based on concept $c$
Require: Teaching heuristic $h(x_i, y_i)$
  while $\mathcal{S}^c \neq \emptyset$ do
      Show example $(x, y) = \text{argmin}_{(x_i, y_i) \in \mathcal{S}^c} h(x_i, y_i)$
      $\mathcal{S}^c \leftarrow \mathcal{S}^c \setminus (x, y)$
  end while
```

2.2.1. *Teaching a binary classifier from a finite example set*

Teaching by selecting examples involves ranking examples in terms of their informativeness for the learner. For instance, in the illustrative problem of teaching a threshold in 1-D (Fig. 1), a useful heuristic would be the distance to the nearest example from the opposite class. This heuristic indeed identifies the two examples closest to the threshold as most informative.

MacGregor [27] proposed two teaching heuristics. The *fast-match* heuristic evaluates each example in terms of the degree that it improves the goodness-of-fit between current and true models.[3] MacGregor observed that this heuristic favors examples that are similar to both categories, while placing *prototypical*, *pure-case* examples lower in the ranking. Based on this observation he proposed the *close-all* heuristic, which ranks examples based on their distance to the decision boundary, as in our illustrative example above. While the fast-match heuristic is impractical as teaching guidance, the close-all heuristic gives an intuitive description of useful examples for a binary discriminative learner. Thus our experiments involving heuristic teaching guidance use this heuristic, referred to as the *borderline* heuristic.

As an example, we consider a simple binary *Nearest Neighbor* (NN) classifier. A NN classifier stores all labeled examples $\mathcal{S}_\tau = \{(x_i, y_i)\}_{i=1}^{|\tau|}$ provided by the teacher. It assigns a label to a new sample x from the state space $\mathcal{X}$ as the label of the most similar example in $\mathcal{S}_\tau$, using a similarity function d over $\mathcal{X}$, *i.e.* the label of x is predicted as y_j, where $j = \text{argmin}_{i=1..|\tau|} d(x_i, x)$. The borderline heuristic is appropriate for this learner, since the ideal operation of a binary NN classifier requires that for every unlabeled sample, the nearest labeled example is in the same class.

In this problem setting, we define the borderline heuristic as $h(x_i, y_i) = \min_{j=1..|\mathcal{S}_c|} d(x_i, x_j) I(y_i, y_j)$ where $I(y_i, y_j)$ is 1 if $y_i \neq y_j$, and infinity otherwise. A computational teacher for this problem can rank examples $(x_i, y_i) \in \mathcal{S}_c$ based on $h(x_i, y_i)$, and present them to the learner in increasing order. Pseudo-code for this algorithm is given in Algorithm 2.

Describing this heuristic to human teachers requires giving them an understanding of borderline examples. A domain specific example of such heuristic teaching guidance is given in Experiment 3 (Section 3.5) for the problem of teaching a binary NN classifier. We note that the optimal teaching algorithm for conjunctions explained earlier in Section 2.1.1 is also consistent with the borderline heuristic, since it provides the negative examples that are closest to being positive (*i.e.* different by a single property). Therefore we hypothesize that the same heuristic teaching guidance will be effective for teaching conjunctions. In Experiment 2 (Section 3.4), we provide the *heuristic* version of the *algorithmic* teaching guidance considered in Experiment 1 (Section 3.3), for the problem of teaching conjunctions.

2.2.2. *Teaching a binary classifier by sample generation*

For some problems there is no existing set of examples ($\mathcal{S}_c$) from which the teacher can choose examples. Instead, the teacher needs to generate examples to be provided to the learner. For instance, a gesture recognition system can be trained by *demonstrating* gestures and labeling them. Ideally, a computational teacher in this setting can directly generate examples subject to a teaching heuristic. This requires the teacher to have an explicit model of the target concept. For example, given a target linear classifier, the teacher can directly generate examples that are close to the decision boundary.

Alternatively, the teaching problem can be converted to the setting where the teacher selects examples from a finite set (Section 2.2.1) by first generating an example set $\mathcal{S}_c$ and then using a teaching heuristic to select examples, as discussed previously. This approach requires a computational teacher to have the ability to generate $\mathcal{S}_c$ by drawing samples x_i from $\mathcal{X}$ and assigning their correct labels y_i.

We note that the intuition behind a teaching heuristic remains the same in both settings. A heuristic that describes properties of informative examples can be used both for *choosing* examples from a set and for *generating* new examples from scratch. Therefore in our experiments we reuse the teaching guidance based on the *borderline* heuristic, in a problem that involves generating examples to teach a binary NN classifier. The domain specific example of this type of teaching guidance is given in Experiment 4 (Section 3.6).

3. Experiments and results

We conducted four experiments to evaluate the effectiveness of the different types of teaching guidance. We first give an overview of these experiments, and then present details and results for each experiment.

[3] Note that this is equivalent to the approximately optimal greedy algorithm discussed in Section 2.1, and it is intractable for human teachers.

3.1. Overall experimental design and procedure

Our experiments are designed to both evaluate natural human teaching and assess the effects of teaching guidance. All are a between-groups design, *i.e.* a participant teaches only one concept in one of the following conditions:

- *Natural*: The teacher is motivated to teach efficiently but not guided about how to teach.
- *Algorithmic*: The teacher is given teaching guidance that is based on an optimal teaching algorithm.
- *Heuristic*: The teacher is given teaching guidance that is based on a computational teaching heuristic.

The first experiment provides a *Natural–Algorithmic* comparison. The second experiment follows up on this experiment with heuristic teaching guidance in the same domain, thus providing a comparison of all three conditions. The last two experiments provide a *Natural–Heuristic* comparison.

All our experiments involve a human teaching a particular concept to a virtual agent. The target concepts are pre-specified and explained to the teacher prior to the teaching session. This is done to ensure comparability of results, but is analogous to a scenario where a human teacher decides to teach their own concepts to an agent. A teaching session involves sequentially selecting or generating instances in the sample space and providing category labels for them. All classification tasks are binary. The teacher is allowed to *test* the learner at any time during the interaction.

All experiments are web-based and share a common webpage layout with three parts separated by horizontal lines. The top part has general instructions to the participant. It motivates the teaching task and describes the concept to be taught. The middle part is a Java applet that lets the teacher select, configure or generate examples, and submit them to the learner. The learner can also be tested with these examples.

The bottom part of the webpage has questions to be answered by the teacher after completing the teaching task. In the *Natural* condition we ask participants two open-ended questions. The first asks them to characterize their teaching strategy and the second asks their criteria for ending teaching. In the *Algorithmic* and *Heuristic* conditions we ask if the teaching guidance was easy to follow and which parts they thought were most challenging.

Participants were solicited and compensated through Amazon Mechanical Turk.[4] Mechanical Turk has been successfully used for user studies [20] and there have been some efforts towards identifying the demographics and other characteristics of its workers [32]. All of our experiments were initially pilot tested with up to five participants to determine potential issues in the interface and instructions. We found through these tests that Mechanical Turk workers were able to interact meaningfully with our interfaces, and that they were able to provide intelligible answers to open ended questions. The responses to the open-ended questions were also used in the actual experiments for filtering. Data from participants who did not provide any open-ended response, or whose responses were not comprehensible were removed and a new participant was recruited to replace it. An individual was allowed to take part only once in any of our experiments, and were compensated \$0.40 or \$0.50 depending on experiment.

3.2. Evaluation

We evaluate teaching with the following measures.

Performance of the taught learner: The performance of the learner during teaching is assessed with the following metrics.

- *Accuracy:* We compute the learner's prediction accuracy throughout the course of teaching, using the complete set of examples as the test set. We present *learning curves* that show how this accuracy changes as examples are sequentially provided to the learner. In addition, we provide the *final accuracy* achieved at the end of the teaching session, to allow a comparison of the teaching performance beyond the first few examples shown on the learning curves.
- *Fraction of questions:* Participants have the option of asking questions to the learner during teaching to test its classification of particular examples. Learning curves do not account for how the teacher exploits this option. To assess the participants' use of questions, we provide the *fraction of questions* among the total number of interactions with the learner, *i.e.* $n_{questions}/(n_{questions}+n_{examples})$.
- *Session duration:* As an additional measure of the teachers' efficiency in teaching, we provide the total duration of teaching sessions (in wall-clock time).

Participant assessment of teaching interaction: We provide statistics about the responses to the open-ended questions, as well as sample quotes from these answers. The responses were coded independently by the two authors, based on the two criteria shown on Tables 2 and 3.

In the rest of this section we go over the four experiments (summarized in Table 1). For each experiment we explain the problem domain, teaching interface and instructions given to participants and we present findings.

[4] www.requester.mturk.com

Table 1
Summary of experiments.

Experiment	Guidance type	Teaching problem	Domain
1	Algorithmic	Conjunctions	FACES
2	Heuristic	Conjunctions	FACES
3	Heuristic	Nearest neighbor (selection)	ANIMALS
4	Heuristic	Nearest neighbor (generation)	GESTURES

Table 2
Number of participants (and the % over all participants) in the *Natural* condition, who partially/fully explained the corresponding teaching guidance when asked to characterize their teaching strategies. The numbers are determined by consensus of the two coders, and initial inter-coder agreement is reported with Cohen's Kappa (κ).

	Exp. 1–2	Exp. 3	Exp. 4
Fully explained guidance	2/80 (2.5%)	2/20 (10%)	4/20 (20%)
Partially explained guidance	18/80 (22.5%)	5/20 (25%)	2/20 (5%)
κ	0.94	0.94	0.97

Table 3
Distribution of participants in the *Algorithmic* and *Heuristic* conditions of all experiments, in terms of whether they had any problems following the teaching guidance.

	Exp. 1	Exp. 2	Exp. 3	Exp. 4
No problems	57/80	30/40	19/20	17/20
Problem understanding	9/80	6/40	0/20	2/20
Problem following	14/80	4/40	1/20	1/20
κ	0.87	0.89	1.00	0.97

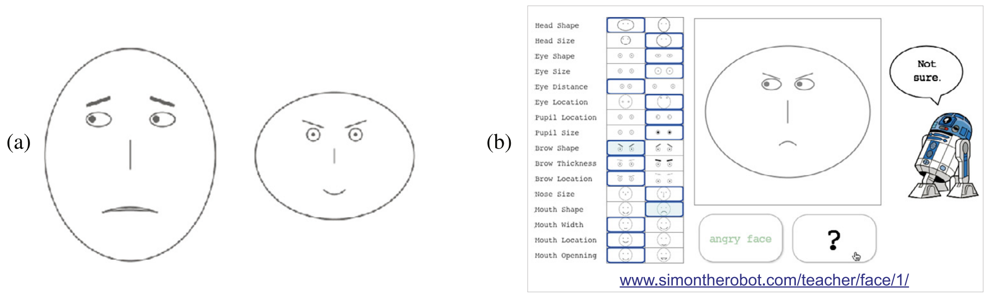


Fig. 2. (a) Two extreme Chernoff faces with 16 binary features in the FACES domain. (b) Interface for teaching conjunctions (16 binary features, *Angry face* concept) with a link to the webpage containing the interface.

3.3. Experiment 1: algorithmic teaching of conjunctions

Our first experiment investigates *Natural* versus *Algorithmic* teaching for the problem of teaching conjunctions, described in Section 2.1.1.

3.3.1. Domain: FACES

We consider conjunctions in the *Chernoff faces* domain [7]. This is a visual representation of multi-dimensional data, where each dimension is a facial feature, and a face instance is a point in this multi-dimensional space. We use faces with 16 binary features. Fig. 2(a) shows two instances with opposite values for all features. A face concept is a conjunction of feature values for a subset of the features. For example an angry face has two relevant feature values: *brow shape* with outer ends raised, and *mouth shape* turned down.

In this experiment, in addition to the teaching mode (*Natural* or *Algorithmic*), we vary the *number of features (n)* of the concept class and the *number of relevant features (r)* of the particular concept (Section 2.1.1). We use two concepts from the class $\mathcal{C}^{16}$ with $r = 2$ and $r = 8$ and two from $\mathcal{C}^{8}$ with $r = 2$ and $r = 4$. Eight features of the face are kept constant for $\mathcal{C}^{8}$.

For learning conjunctions from example, the learner uses the halving algorithm based on version spaces [26]. Unique identification corresponds to being left with a single hypothesis in the version space. In order to have an accuracy that is consistent with unique identification, the learner responds to a test only when the whole version space agrees on its label. Otherwise the learner will say "Not sure" even if the majority of the version space predicts one label.

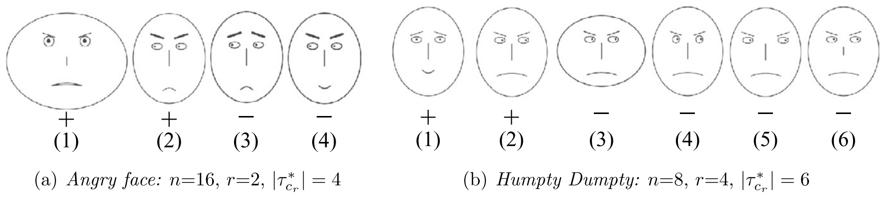

+
+
−
−
(1)
(2)
(3)
(4)
(a) Angry face: n=16, r=2, $|\tau^*_{c_r}| = 4$
+
+
−
−
−
−
(1)
(2)
(3)
(4)
(5)
(6)
(b) Humpty Dumpty: n=8, r=4, $|\tau^*_{c_r}| = 6$

Fig. 3. Example optimal teaching sequences for 2 conjunction concepts in the *Chernoff faces* domain.

3.3.2. Interface

A screenshot of the teaching interface is shown in Fig. 2(b). The applet has all facial feature names listed on the left, with the two possible values of each feature shown as icons. At any time one of the two feature values is *selected*, indicated by a blue square around the corresponding icon. The teacher can change the selected value of a feature by clicking on the icon of the opposite value. The selected features are randomized when the applet is first loaded. The face that corresponds to the currently selected list of feature values is depicted at the center of the interface. Relevant features of the target concept are highlighted as light blue to help the teachers.

There are two buttons below the face. One lets the teacher submit the current face as an example to the learner. The label of the example is automatically changed to the correct label. The second button allows the teacher to test the learner with the currently configured face. The learner is depicted as a robot, and responds to questions with a speech bubble.

3.3.3. Instructions and teaching guidance

The instructions motivate the teaching task with a scenario: Participants are told that a robot, R2D2, will go to a planet to deliver medication to certain humans that can be recognized by facial characteristics. Then, the n facial features and the particular face category (c_r) that the robot needs to learn are described.

In the *Natural* condition, participants are motivated to teach optimally. They are told that R2D2 will be tested after the study, and that the participant who trains the best performing robot, with the least number of examples will receive extra compensation. In both conditions, participants are told to stop teaching when they think the robot has learned the concept. In the *Algorithmic* condition, participants are asked to follow the provided teaching guidance based on the algorithm given in Section 2.1.1. As an example, the teaching guidance for teaching *angry face* is as follows:

- ***Step 1**: Configure one angry face example and show this to R2D2.*
- ***Step 2**: Starting from the example from Step 1, change all properties that don't matter for angry face to the opposite value. This results in another angry face maximally different from the previous one. Show this example to R2D2.*
- ***Step 3**: Starting from the example from Step 2, change one of the properties that matter for angry face to the opposite value. This results in a non-angry face that is minimally different from an angry face (by just one property). Show this example to R2D2.*
- *Change the property from Step 3 back to the right value. Then repeat Step 3 for other properties that matter for angry face. So, show different examples of NOT angry face which differ from an angry face by just one property.*

These instructions are not specific to *angry face*. The guidance for the second face used in the experiment, *Humpty Dumpty*, is exactly the same other than the category name. Fig. 3 shows example optimal teaching sequences produced with the algorithm described in Section 2.1.1 for both concepts.

3.3.4. Results

This experiment was completed by 160 participants, 20 in each condition.[5] Fig. 4 compares the average accuracy achieved with the example sequences provided in the *Natural* versus *Algorithmic* conditions across the four target concepts.[6] For comparison, learning curves of the optimal teaching sequences (Section 2.1.1) is also shown. We make the following observations.

Natural teaching is rarely optimal. In the *Natural* condition most teachers failed to teach the concept optimally. Only 17 out of the 80 participants were eventually able to provide sets that uniquely identify the target concept. There was one instance of optimal teaching, and from the participant's description of his teaching strategy, we see that their optimality was not accidental; the description matches the optimal teaching algorithm:

> "Showed him examples of everything changing but the key elements that create the angry face, then kept all of the other variables constant and just changed the parts that make the face not angry or angry to show the difference between the two."

[5] The eight conditions are due to the three factors varied in the experiment: teaching mode (*Natural* versus *Algorithmic*), size of concept space (n) and concept size (r).

[6] The reason that accuracy starts at zero is that the learner respond to tests with "I don't know" when it is not 100% certain (see Section 3.3.1). Zero accuracy does not mean that the learner classifies all instances wrong. It means that it classifies nothing correctly. Due to this behavior, the learner never makes a mistake when it responds to a test.

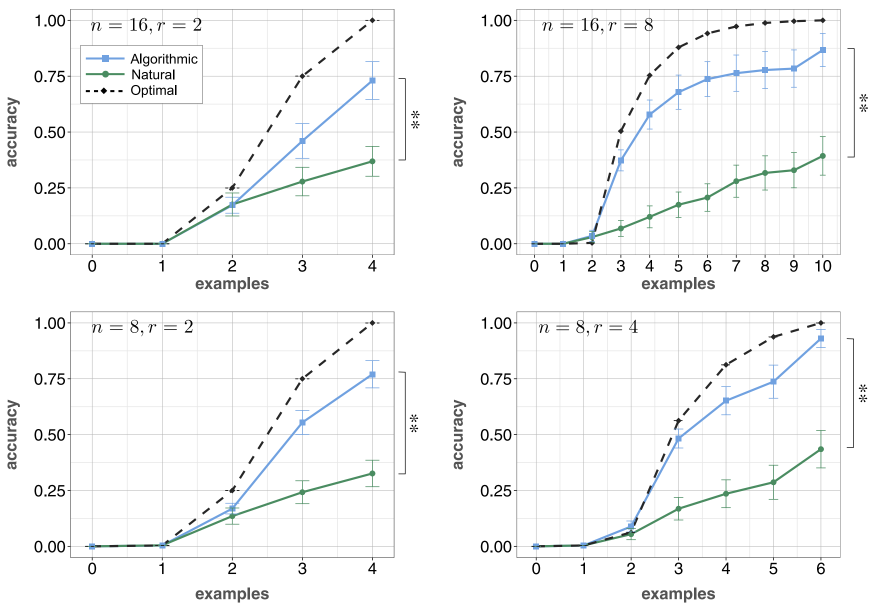


Fig. 4. Results for Experiment 1: Progression of the average accuracy on the complete sample space, over the first $TD(c_r, C^n)$ examples while teaching four different target conjunctions in the FACES domain.

As shown in Table 2 one other participant's described strategy matches the algorithmic teaching guidance, though their teaching sequence was not optimal. In addition we see that 18 participants in the *Natural* condition partially described the optimal teaching algorithm, where 'partially' is defined as giving two or more steps of the algorithm. An example is the following:

"I showed him the correct combination first, then changed each correct feature one by one, until I had shown him all the incorrect combinations."

This shows that even though the optimal teaching sequence is unlikely to occur spontaneously without teaching guidance, humans have an intuition for what examples would be informative to the learner. A different strategy that was observed in this experiment is presenting examples only when the learner responds "Not sure" when tested with it. 16 participants in the *Natural* condition mentioned using tests as part of their strategy. Such tests can allow the teacher to build a better mental model of what the learner knows and how it learns. Consequently, the teacher presents more informative examples.

Teaching guidance improves teaching accuracy. As seen in Fig. 4, the algorithmic teaching guidance results in learning curves that are much better than the naturally occurring ones, and that are close to optimal. In the *Algorithmic* condition, 41 participants out of 80 uniquely identified the target concept with their examples, 17 of them being optimal. The average accuracy after $|\tau^*_{c_r}|$ examples is 0.82 (SD = 0.31) as compared to a 0.39 (SD = 0.32) in the *Natural* condition ($t(158) = -7.98$, $p < .001$, in a student's t-test). This difference persists after $|\tau^*_{c_r}|$ examples; as seen in Fig. 5 the average accuracy is in all cases higher in the *Algorithmic* condition at the time when the participants decide to end teaching.

Teaching guidance reduces teaching time. The algorithmic teaching guidance makes human teachers more efficient. In addition to the improved accuracy, the total time spent teaching is smaller in the *Algorithmic* condition, as seen in Fig. 5. Besides the higher number of examples provided in the *Natural* condition, the higher number of questions also contributes to longer session durations. As mentioned earlier, a common strategy in this condition was to extensively test the learner and provide examples when it responds indecisively, which appears to be a time consuming strategy.

Parts of teaching guidance are less intuitive. Even though guidance improves learning as compared to the *Natural* condition in general, it could be followed perfectly by only 17 participants out of the 80. The rest of the participants had some problem following the strategy. In the open-ended question that asked participants whether they had any problems following the provided guidance, a majority of the participants (57 out of 80) said that they had no problems. This means that some participants made mistakes while applying the guidance, without realizing it. 14 participants expressed difficulties in following parts of the guidance, all of which pointed to the part of the guidance related to giving negative examples (*i.e.* starting at Step 3 of the guidance given in Section 3.3.3). Some examples are "While repeating step 3 readjusting the face to the original state was difficult" or "It was a little hard to remember which ones had I switched already".

Teaching optimally is harder in larger state spaces. Fig. 4 shows that the learning curves for the *Guided* condition are closer to optimal for $n = 8$, for both values of r. This trend indicates that algorithmic teaching guidance, which involves manipulating features one by one, might become intractable for humans as the state space gets larger.

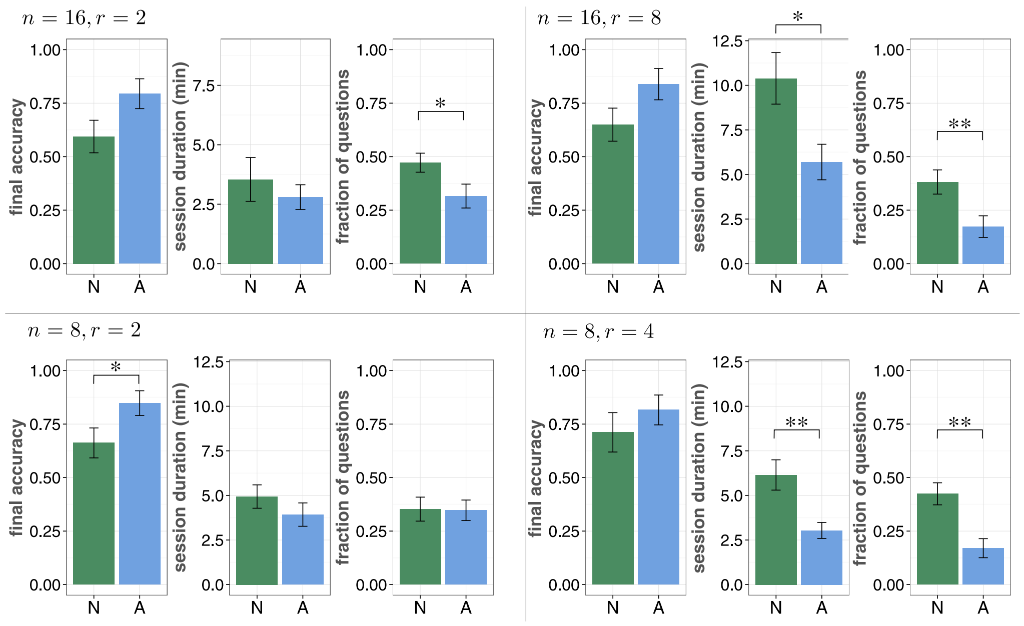


Fig. 5. Additional results for Experiment 1: Final accuracy, session duration and fraction of questions.

Teaching guidance has more impact on longer teaching sequences. When r has a larger value with respect to n we see a larger difference between the learning curves of the *Natural* and *Algorithmic* conditions (Fig. 4: Right column). We also see that the difference between the two conditions is less significant in terms of the time spent teaching and the number of questions asked during teaching, in the two sub-conditions that involved shorter optimal teaching sequences (Fig. 5: Right *versus* Left column).

3.4. Experiment 2: heuristic teaching of conjunctions

In the second experiment we consider *heuristic* teaching guidance for the same problem studied in Experiment 1. We posit that the *borderline* heuristic captures the intuition of the optimal teaching algorithm for conjunctions. Therefore, we explore the use of this heuristic as teaching guidance. We consider this experiment as a follow-up on Experiment 1 and compare the results from the *Heuristic* condition in this experiment, with both *Natural* and *Algorithmic* condition in Experiment 1. The domain and the interface are the same as Experiment 1.

3.4.1. Instructions and teaching guidance

The instructions in the *Heuristic* condition, are based on the borderline heuristic. For example, the guidance for teaching *angry face* is:

- *When you show examples of **Angry face** vary them as much as possible. Show angry faces that are very different from one another.*
- *When you show examples of **NOT Angry face** make your examples as similar to Angry face as possible, but different in some aspect. Show examples that are almost Angry face.*

In addition we remind the participants to alternate between positive and negative examples as they like, in order to avoid them interpreting the instructions as being sequential.

The *heuristic* teaching guidance provides less information than its *algorithmic* counterpart. Although it correctly specifies each example in an optimal teaching sequence (*e.g.* Fig. 3), it does not specify the number and order of positive and negative examples to be provided. Therefore we expect *heuristic* guidance to be less effective than *algorithmic* guidance; however, it will still provide significant improvements over unguided, natural teaching.

3.4.2. Results

This experiment was completed by 40 participants, 20 in each sub-condition where $r = 2$ versus $r = 8$ (with $n = 16$ kept constant). These two sub-conditions are comparable with the first two sub-conditions of Experiment 1. Learning curves for accuracy and additional results on final accuracy, duration of sessions and the fraction of questions during teaching are shown in Fig. 6. We make the following observations.

Heuristic teaching guidance improves teaching. Heuristic teaching guidance results in better learning curves than naturally given teaching sequences over the first $|\tau^*_{c_r}|$ examples (Fig. 6). It also results in better final accuracies at the end of

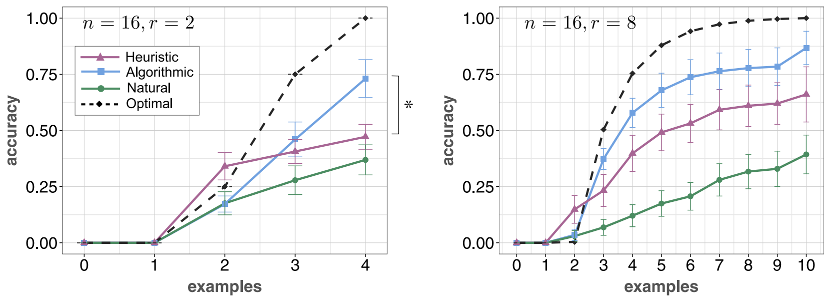


Fig. 6. Results for Experiment 2: Learning curves for teaching conjunctions with heuristic teaching guidance, compared to learning curves from Experiment 1, in terms of accuracy on the set of all possible faces.

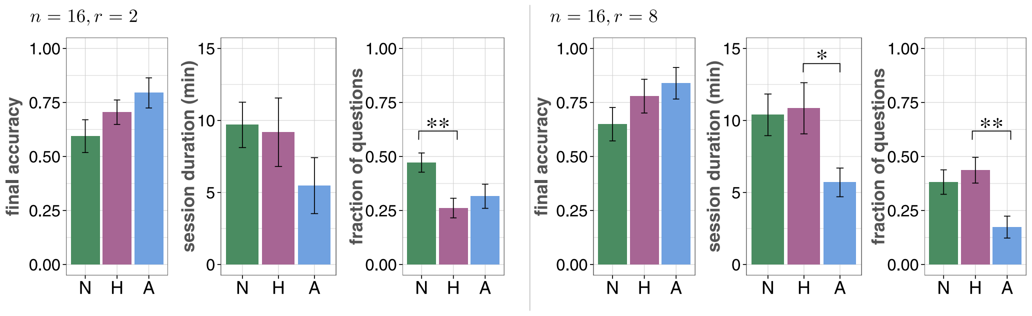


Fig. 7. Additional results for Experiment 2: Final accuracy, session duration and fraction of questions.

teaching sessions (Fig. 7). A majority of the participants (30 out of 40) reported that they had no problem understanding or following the heuristic teaching guidance. As the most challenging part of the guidance, participants alluded to the lack of steps — for example, “how many different faces were we supposed to show R2D2?”, “It is not clear when the task ends.” and “I wasn’t sure whether or not I had to switch between yes humpty/no humpty”. This points to one of the fundamental differences between heuristic and algorithmic teaching guidance.

Heuristic teaching guidance is not as effective as algorithmic. We find that the learning curves obtained with *heuristic* teaching guidance are not as good as the ones obtained with *algorithmic* teaching guidance in Experiment 1. As noted in the previous paragraph, this was mainly due to a lack of instruction on how to use the given heuristic as part of a teaching algorithm. The participants had to decide for themselves when and how many positive and negative examples they would present to the learner. The heuristic helped them only with choosing good positive or negative examples. While this provided improvements over the *Natural* condition, it was not sufficient. This indicates that *algorithmic* teaching guidance is preferable over *heuristic* when it is available.

Teaching guidance has more impact on longer teaching sequences. As with the algorithmic teaching guidance in Experiment 1, we observe that heuristic teaching guidance is more effective in getting humans to provide more useful examples in the sub-condition with $r = 8$, where the optimal teaching sequence is longer (Fig. 6). However, we do not see improvements in terms of the time used or the number of questions asked to the learner. On the contrary, we see that participants spent slightly more time and asked more questions than in the *Natural* condition when $r = 8$ (Fig. 7: Right). This can be attributed to the aforementioned problems encountered by participants in following *heuristic* teaching guidance. These problems were more prominent when teaching required more examples.

3.5. Experiment 3: heuristic teaching by example selection

In the third experiment we move to a problem that does not have a known efficient optimal teaching algorithm. We consider teaching a binary nearest-neighbor classifier by selecting examples from a finite set of examples, as discussed in Section 2.2.1.

3.5.1. Domain: ANIMALS

We consider the classification of images containing animals into two categories: *polar bear* and *other animals*. Our dataset has 25 instances in each category randomly sampled from the *Animals with Attributes (AwA)* dataset [24]. The set of other animals includes *pandas*, *giraffes*, *raccoons*, *seals* and *wolves*. Sample images are shown in Fig. 8(a).

For classification of the images we used a set of color histogram features. Although the *AwA* dataset has a number of other features such as SIFT, PHOG and SURF, we chose to only use color features since they can be intuitively described to human teachers. Each image is represented by a 2688-dimensional feature vector, with 128-dimensional color histograms

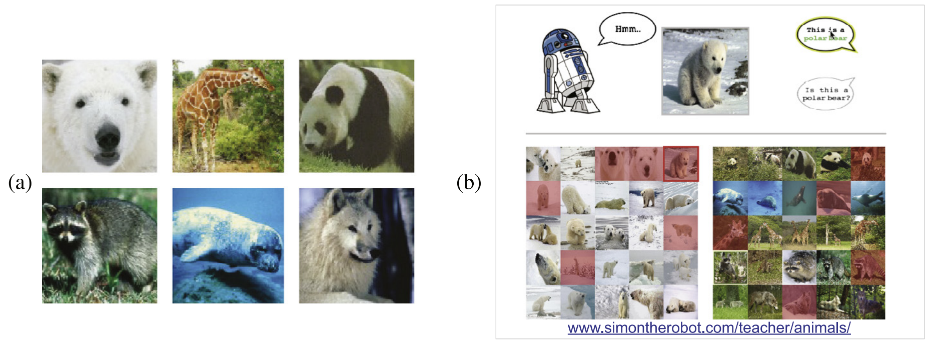


Fig. 8. (a) Sample images from the ANIMALS domain. (b) Interface for teaching a binary nearest-neighbor classifier (*polar bears* versus other animals) by selecting examples from a pool.

for 21 grid cells at three different scales $(1 + 4 + 16)$. The nearest-neighbor classifier uses L_1-norm of these feature vectors as the distance between two images.

3.5.2. Interface

The applet contains two groups of 25 image thumbnails, respectively containing images of *polar bear* and other animals (see Fig. 8(b)). One of the 50 thumbnails is selected at all times and is displayed at the top. The selected thumbnail is indicated by a red rectangle around the thumbnail, and the user can select a different image by clicking on a different thumbnail. The initially selected image is chosen randomly when the applet is loaded. To the right of the selected image display, there are two buttons that allow the participant to label the image or ask the learner for its prediction of the label. When an image has been labeled, its thumbnail becomes highlighted in red, and it cannot be labeled again.

3.5.3. Instructions and teaching guidance

The instructions tell the participants that they will train a robot, R2D2, to go to a planet to deliver medication to polar bears. It says that R2D2 recognizes an animal by taking its picture and analyzing the color content and distribution of that picture. Therefore they are asked to teach R2D2 by showing it pictures of animals; both polar bears and other animals.

As in the other experiments, in the *Natural* condition, teachers are motivated to teach as well as they can. In the *Heuristic* condition, participants are asked to follow the provided teaching guidance that is based on the *borderline* heuristic discussed in Section 2.2.1:

- *When you show examples of **polar bear** make them as different from one another as possible. Try to show examples of polar bear that are easier to confuse with other animals.*
- *Similarly, when you show examples of other animals which are **NOT polar bears**, make them as different from one another as possible, and show examples that are easier to confuse with polar bear pictures.*

3.5.4. Results

This experiment was completed by 40 participants, 20 in each of the two conditions: *Natural* versus *Heuristic*. In Fig. 11 the learning curves for the two conditions are compared with the approximately-optimal greedy teaching algorithm, and a heuristic teaching algorithm that uses the borderline heuristic. In this experiment, we constrain the heuristic teaching algorithm to provide a *balanced* set of positive and negative examples. Different from Algorithm 2, this algorithm ranks the positive and negative examples separately based on the borderline heuristic, and presents them to the learner in increasing order, alternating between positive and negative examples. Additional results on the final accuracy, duration of sessions and the fraction of questions during teaching are also provided in Fig. 11.

The borderline heuristic is effective. Before looking into the use of the heuristic as teaching guidance, we examine the performance of the computational borderline heuristic described in Section 2.2.1. As seen in Fig. 11(a), the heuristic is effective in choosing examples that result in a learning curve that quickly converges to the approximately optimal teaching sequence. Note that in this case, the approximately optimal teaching sequence is actually optimal, due to the distribution of the data set. In other words, there exist two examples (one from each class) such that all positive examples are closer to the positive one, and all negative examples are closer to the negative one; and the greedy teaching algorithm selects these two examples as the first two. The heuristic algorithm is not able to select these examples immediately since these are not necessarily the most "borderline" examples; however, providing examples chosen by the heuristic results in a close approximation that is overall better than the examples provided by people in this experiment.

To give a better intuition about the operation of the heuristic teaching algorithm, we further examine the examples selected based on the borderline heuristic. Fig. 9 gives a visualization of the distribution of all examples in the used dataset projected onto a 2D plane using Isomap [40]. The examples that appear to be borderline in this visualization are intuitive. For instance, examples of *non-polar bear* images that are closest to the border involve different animals (*raccoon*, *wolf* and

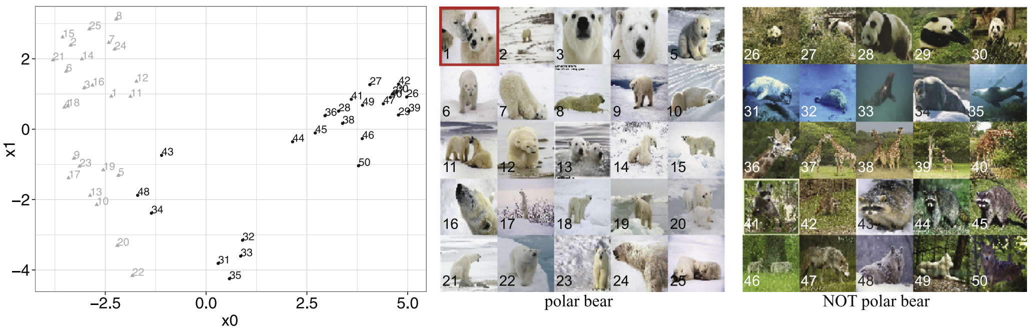


Fig. 9. Visualization of the distribution of examples in the ANIMALS domain with Isomap and the complete set of examples used in Experiment 3.

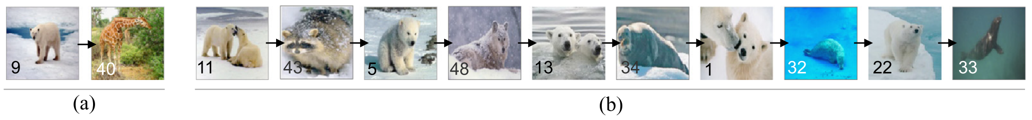


Fig. 10. Teaching sequences produced by computational teachers: (a) greedy teacher, (b) heuristic teacher using the borderline heuristic. Numbers indicate the index of the image on Fig. 9.

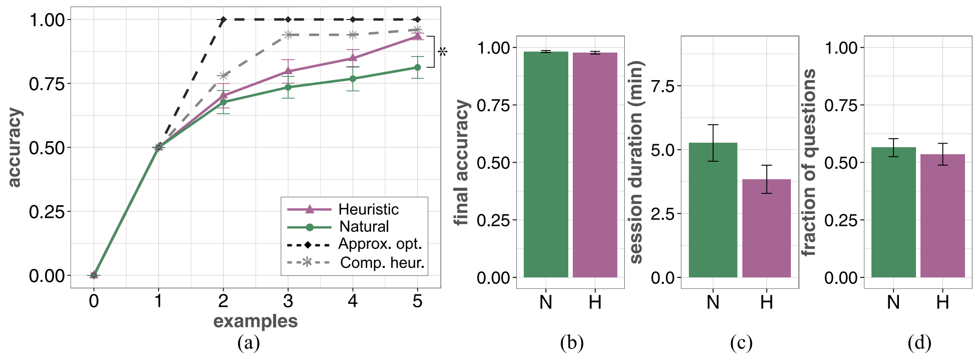


Fig. 11. Results for Experiment 3: (a) Accuracy over the first few examples when teaching a binary classifier in the ANIMALS domain, by selecting example images from a pool. (b) Final accuracy, (c) session duration and (d) fraction of questions.

seal) in a snowy background, the typical polar bear setting. Fig. 10 shows the first ten examples selected by the heuristic teaching algorithm. These are indeed examples that appear to be close to the border with the opposite class in the visualization from Fig. 9 and are consistent with a human understanding of borderline.

Heuristic teaching guidance improves accuracy. We find that the learning curve is closer to the one of the approximately optimal sequence when the participants receive heuristic teaching guidance. This difference is significantly different after the first five examples ($t(38) = -2.09$, $p = .046$), and the performance of the learners taught with heuristic guidance is very close to the performance of the heuristic teaching algorithm, and almost perfect. However, we see that this difference between the two conditions eventually vanishes, as the final accuracy of the learners at the end of teaching sessions are both close to perfect (Fig. 11(b)). This is mainly due to the small sample size (total of 50 examples) making it possible to show sufficient examples for perfect classification, within the time that participants are willing to allocate to the experiment. Nonetheless, participants that receive teaching guidance seem to be more efficient in achieving this, using less time (Fig. 11(c)) and requiring less examples.

Looking closer at the examples chosen by participants, we see that their interpretation of the teaching guidance is in line with the borderline heuristic. Fig. 12 contrasts the examples chosen by participants with and without guidance. The positive examples in the *Heuristic* condition are more evenly distributed, rather than being concentrated on a few prototypical examples as in the *Natural* condition. In addition, more participants in the *Heuristic* condition pick borderline negative examples which appear in the teaching sequence produced by the computational heuristic teacher. In particular, they provide more examples of different animals in a snowy background (Fig. 12(a)), and more examples of *seals* where the background involves the blue color of the sea (Fig. 12(b)), which also occurs in several of the *polar bear* images. We also see that participants in the *Heuristic* condition provided more examples of a particular *panda* image (Fig. 12(c)). While this example was not borderline in terms of the features represented by our learner, it is not surprising for humans to think of *polar bears* and *panda bears* as similar. This means our description of the learner's features was effective in allowing

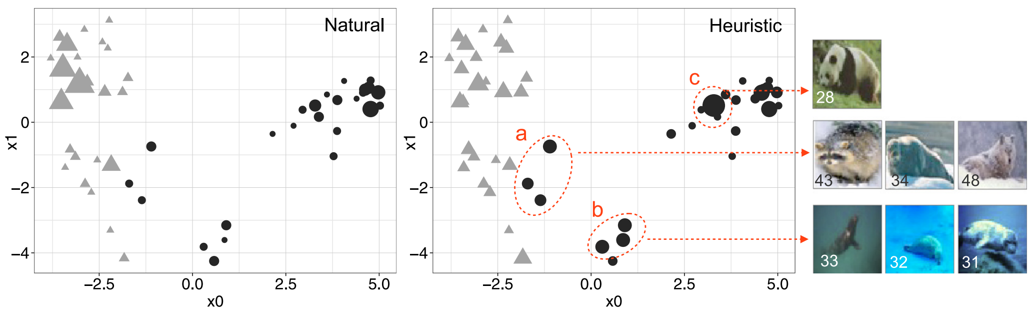


Fig. 12. Differences in the choice of examples between the *Natural* and *Heuristic* conditions in Experiment 3: size of the marker is proportional to the frequency of usage of the example as one of the first five examples in the teaching sequence.

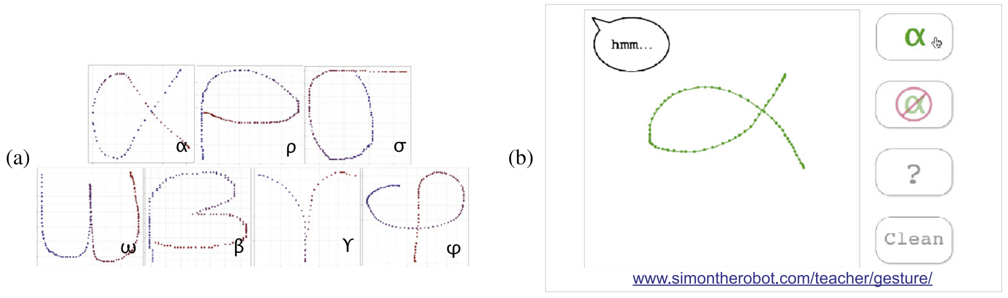


Fig. 13. (a) Examples from data collected for the GESTURES domain. (b) Interface for training a binary nearest-neighbor classifier (alpha versus other gestures) by producing gestures and labeling them.

participants to pick borderline examples; however, it did not stop them from providing borderline examples in terms of additional features not in the learner's representation.

3.6. Experiment 4: heuristic teaching by example generation

Our final experiment investigates *Natural* versus *Heuristic* teaching of a binary nearest neighbor classifier by *generating* examples, rather than selecting them from a pool of existing examples.

3.6.1. Domain: GESTURES

We consider the scenario of training a computer mouse gesture recognition system. We represent a mouse gesture as a time series of 2D points. Gestures can be of varying length. As a dissimilarity measure between two gestures, we use the warping distance obtained from applying dynamic time warping (DTW) between two gestures. DTW measures the similarity between two sequences by optimally aligning them in time and considering the distances between aligned pairs of points in the sequences. We consider gestures for drawing Greek letters (Fig. 13(a)), and focus on the binary classification of whether a gesture is *alpha* (α) or not.

To evaluate this teaching problem we collected a separate test corpus for seven gestures from 20 participants (different from participants in the user study). All participants provided 5 examples for each gesture. Sample gestures are shown on Fig. 13(a).

3.6.2. Interface

The interaction interface is shown in Fig. 13(b). It has an area for drawing the gesture, and the trace of the gesture is visualized as it is being drawn. Two buttons let the teacher submit the drawn gesture as an example of α or not-α. A third button tests the learner with the current gesture. A fourth button cleans up the drawing area. The gesture is automatically cleaned up when submitted as an example, or when another gesture is performed on the drawing area. Therefore, each gesture is a single movement.

The interface for collecting the test set is similar. There is only one button for submitting the drawn gesture. The button indicates the gesture that is to be drawn, which is one of the seven Greek letters. After five examples of each letter the button automatically changes to the next letter.

3.6.3. Instructions and teaching guidance

The instructions tell the participant that there are seven possible gestures corresponding to drawing Greek letters with the mouse. They are told to teach the system to recognize whether a gesture is alpha or not.

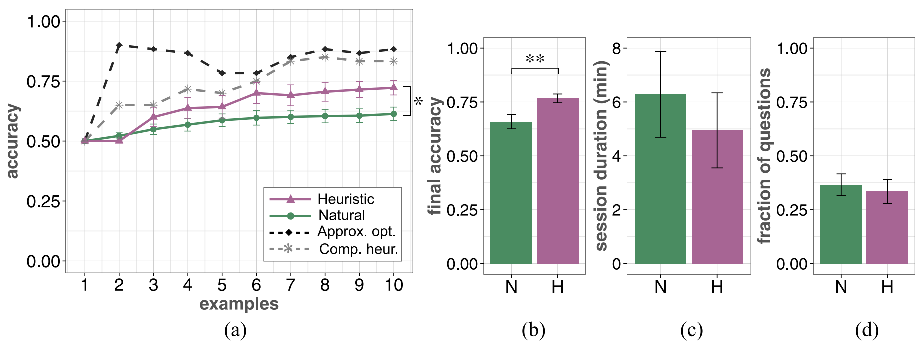


Fig. 14. Results for Experiment 4: (a) Accuracy of the binary nearest-neighbor classifiers (GESTURES) over the first few teaching examples compared to the teaching sequences produced by the approximately optimal greedy teaching algorithm and a heuristic teaching algorithm that uses the computational *borderline* heuristic. (b) Final accuracy, (c) session duration and (d) fraction of questions.

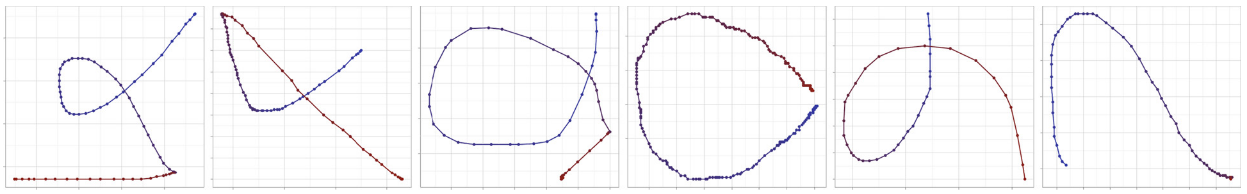


Fig. 15. Examples of non-alpha gestures given by participants, which illustrate the use of the *borderline* heuristic in the *Heuristic* condition of Experiment 4.

To motivate good teaching in the *Natural* condition participants are told that the system that they train will be tested on a separate set of gestures and that the teacher of the best performing system will receive extra compensation. In the *Heuristic* condition the borderline heuristic is explained to the participant in the form of guidelines for choosing examples. The guidance is worded as follows:

- *When you show examples of **alpha** gestures, vary them as much as possible. Try to show alphas that are borderline examples (i.e. almost not-alpha).*
- *When you show examples of **not alpha** gestures, try to make them quite similar to alpha. Focus on the gestures that look most like alpha and have a similar structure with alpha. In other words, make your not alpha examples borderline examples which are near-misses.*

3.6.4. Results

Our final experiment was completed by 40 participants (20 in each condition). Learning curves corresponding to the first few examples provided by participants in the two conditions are given in Fig. 14. The learning curves are in terms of the accuracy on the collected data set described in Section 3.6.1. We also compare both conditions to the performance achieved by an approximately optimal teaching algorithm and a heuristic teaching algorithm that balances positive and negative examples (described in Section 3.5.4). Note that, the computational teaching algorithms used for comparison do not generate examples from scratch; they pick examples from the collected data set. We make the following observations.

The borderline heuristic is effective. As in Experiment 3, the heuristic algorithm that uses the borderline heuristic gives close performance to the approximately optimal greedy algorithm, especially after the few initial examples. We note that unlike the dataset in Experiment 3, the data obtained for the GESTURES domain is more noisy and not separable. As a result the greedy algorithm results in a more bumpy learning curve.

Heuristic teaching guidance improves accuracy. The heuristic teaching guidance helps participants provide better gesture examples. The learning curve obtained when participants receive teaching guidance is closer to the ones produced by the approximately optimal and heuristic algorithms. Note that, these computational algorithms choose examples directly from the test pool, while the participants generate new gestures to train the learner. This means that the computational teachers have an added advantage of having access to the test pool. Therefore, our results demonstrate that teaching heuristics can be effective as teaching guidance, even when the sample needs to be instantiated from scratch in a high dimensional space.

We also see that the final accuracy when the participants end the teaching session is significantly higher in the *Heuristic* condition (Fig. 14(b)), and participants take less time on average in this condition (Fig. 14(c)).

By visually inspecting the examples given by human teachers in the *Heuristic* condition, we see that the *borderline* heuristic is employed by the participants in different ways. Fig. 15 shows a few examples of non-alpha gestures given in this condition. We see that people interpret the notion of borderline-negative as examples that differ from a positive example by a *few* and *relevant* features. The shown non-alpha gestures give some examples of features that are important for a gesture to be considered *alpha* from the human's perspective. Although these features are not explicitly represented by the learner used in our experiments, these examples still provide improvement over natural teaching.

Nevertheless, we find that some participants in the *Natural* condition realized the importance of borderline examples. For instance, one participant describes his strategy as follows:

> "Give the machine a few examples of alpha, then a few examples of obviously not alpha, then test close to but not quite cases of alpha in order to fill the gap."

This demonstrates that the heuristic is intuitive for human teachers; however, it is less likely to be used without explicit guidance.

4. Discussion

Generality of teaching guidance. We are interested in the interactive ML scenario of a human training an agent to perform a classification task by showing examples. While the concepts that humans might want to teach their agents can vary arbitrarily, what constitutes an *informative example* for a particular learner remains constant. In this article, we demonstrated that instructions given to a teacher about what informative examples are for binary classifiers in different domains, positively impacted the examples that they presented to the learner. This guidance is not specific to a particular concept; it is specific to a particular *concept class* and *learning algorithm*. Hence, teaching guidance needs to be specified only once for a particular learner.

Our paper explored teaching guidance for different binary classifiers in representative domains. We expect similar findings with other discriminative learners in different domains; however, the interaction between the concept class, learning algorithm, and domain is complex and more work is needed to make stronger claims. Furthermore, for different problem types (*e.g.* multi-class classification, regression, reinforcement learning) more research is needed to theoretically understand optimal algorithmic teaching, so as to inform teaching guidance for learners in these settings. In a different article, we developed an approximately optimal computational teacher for an inverse reinforcement learning agent in the sequential decision making problem setting, and demonstrated that heuristic teaching guidance based on this teacher had similar effects as the ones studied here [5].

Sub-optimality of natural human teaching. We saw that humans are usually not spontaneously optimal for the range of learners used in our experiments. This is in part due to not knowing how the learner learns. We believe that explaining how the learner works to the participants can improve their inputs to the system. However, in the scenarios considered here, this would be highly inefficient. Explaining concepts like version spaces or L_1-norm would require significant time and effort. In addition, inferring the optimal teaching algorithm from this understanding would still be challenging for everyday users, since it is so even for a theoretician.

Another reason could be anthropomorphization, *i.e.* assuming that the learner learns like a human would. For instance, one common strategy in the first experiment was changing features one-by-one rather than multiple irrelevant ones all at once. This makes sense since it might be difficult for a human learner to pay attention to *everything* that changes between two examples if multiple things changed. Similarly, in the other experiments giving *typical*, *obvious* or *easy* examples first, rather than *borderline* or *difficult* ones, was common. This is consistent with the curriculum teaching approach which was also found to be more prominent in human teaching in previous studies [19,30,4]. One approach is to make learning agents that better accommodate these common teaching patterns in humans [42,33], whereas our research investigates the extent to which we can influence human teaching to accommodate how an agent learns.

Our claims about the sub-optimality of natural or spontaneous teaching are only *with respect to* the particular learners used in our experiments. The examples provided by the participants are possibly optimal for *some* learner. Our research here is about the extent to which their natural mental model matches an arbitrary learner and whether they can easily be guided to provide more optimal teaching sequences for that particular learner.

Limitations of algorithmic teaching guidance. Our first experiment deals with a learner for which there is a known optimal teaching algorithm. We derive the teaching guidance directly from this algorithm. Therefore if the participants followed the guidance perfectly, they would be optimal teachers. While some participants were able to do this, a large portion were still not fully optimal teachers. Participants were often successful at providing the positive examples but failed at providing the negative ones that varied by one feature at a time. As indicated in their open ended answers, this was mainly due to the difficulty of remembering the original positive examples (so they can make sure the negatives differ by only one feature) and remembering which negative examples have already been shown. Again, the participants interpreted the teaching guidance correctly, but their memory and attention limitations made it difficult for them to follow it perfectly. This could be addressed with an improved interface that stores previous examples and allows configuring the current example starting with previous examples.

We believe that having an *active* learner can help with some of the difficulties encountered while following teaching guidance. For example, in the case of conjunctions, we saw that the steps in which the teacher needs to provide multiple negative examples by changing one feature at a time was hardest to follow for human teachers (Section 3.3.4). This points towards a *mixed-initiative* learning approach that combines teaching guidance with active learning [6]. In this problem, an uncertainty-based strategy for an active learner is to make queries by changing one feature at a time, once given a positive seeding example. This allows the learner to test the relevance of each feature independently. Therefore an ideal mixed-initiative approach would be to ask the teacher to provide the first two positive examples that differ in all irrelevant features and then let the learner make queries with the uncertainty-based strategy. This combination would still be optimal, since the teacher would be implementing the first two steps of the optimal teaching algorithm, and the queries made by the learner would be equivalent to the rest of the algorithm. This would also reduce the amount of guidance given to the teacher to the parts that were most intuitive for them. Similarly, in the other domains, by giving better examples than they normally would, the guided teacher would be seeding an active learning algorithm such that the space that the learner needs explore is much smaller than it would be if seeded with random examples.

Limitations of heuristic teaching guidance. In Experiments 3 and 4 we show that for learners with no theoretically optimal teacher, we can supply human teachers with a heuristic. In both experiments, the *borderline* heuristic leads them to provide more informative teaching sequences for the learner. This heuristic is applicable to a wide range of problems—for most discriminative learners, examples that are close to the decision boundary will be most informative. The two particular problems considered in this article are two representative problems involving example selection and example generation. Many practical problems that currently employ ML fall into either of these types, and can benefit from more informative datasets. For instance, consider training a spam classifier which involves selecting examples from your mailbox. When showing examples of emails that are *not* spam, it would be more useful for the learner to receive examples that risk being classified as spam, rather than examples that are obviously not spam.

The generality of teaching heuristics also extends to problems where there are known optimal teaching algorithms. The borderline heuristic is a generalization of the optimal teaching algorithm for conjunctions. However, as demonstrated in Experiment 1 and 2, there is a trade-off between the generality and the effectiveness of teaching guidance. The heuristic teaching guidance is not as effective as the algorithmic teaching guidance for teaching conjunctions, but the algorithmic teaching guidance is only applicable to conjunctions. Thus, when algorithmic teaching guidance is available for a particular learner, it is preferable to use that instead of the heuristic teaching guidance.

This is further supported by the participants' assessment of the difficulty of following teaching guidance. Almost no participants in Experiments 3 and 4 indicated that they had difficulty, whereas several participants in Experiment 2 complained about the lack of order information or ending criteria (Table 3). This indicates that heuristic guidance was less intuitive for the problem that has an exact optimal teaching algorithm.

5. Related work

As noted in Section 1, our work is closely related to Algorithmic Teaching, and we reviewed concepts from this field that are used in our work throughout Section 2. In this section we situate our work with respect to related works from Cognitive Science that study human teaching and learning, then overview work on ML systems with a human in the loop.

5.1. Human teaching and teaching humans

Human teaching has been studied experimentally on different teaching tasks and a number of theoretical accounts and formalisms have been proposed. The *pedagogical sampling* framework introduced by Shafto and Goodman [33] assumes a benevolent teacher and learner. In this framework, as in the Teaching Dimension model of Goldman and Kearns [15], the teacher chooses data that will be most informative for the learner in inferring the target hypothesis. The learner tries to infer the correct hypothesis assuming that the teacher is benevolent. Shafto and Goodman [33] present an experiment in which participants teach axis-parallel rectangles by showing a specified number of examples. They find a strong correlation between the examples predicted by pedagogical sampling and the examples chosen by human teachers. Note that this does not imply humans were optimal teachers in these experiments, because the examples given by the participants are compared to specific pedagogical sampling cases (*e.g.* the most informative examples given that there are two positive examples), rather than a globally optimal teaching sequence for axis-parallel rectangles. However, the strong correlation suggest that the examples chosen by humans were highly informative to the learner. Warner et al. [44] extends the framework to cases where the benevolent teacher assumption is removed and the intent of the teacher needs to be inferred by the learner.

Khan et al. [19] performed an experiment with the illustrative teaching problem shown on Fig. 1, *i.e.* teaching a threshold in 1D. They characterize three teaching strategies used by humans: *linear* (starting from one end of the range and moving towards the other linearly), *positive-only* and *extreme* (starting from the ends of the range and alternating between classes). They observe that most humans use the *extreme* strategy. Consistent with our observation that spontaneous optimal teaching is rare, they observed no instances of a *boundary* strategy. They present a theoretical account of the predominant strategy based on the assumption that humans represent examples in high dimensional spaces rather than just their 1D projection. This justifies the presented example sequences as most informative. The teaching strategy of starting with easy examples (far from the boundary) and gradually increasing the difficulty of examples (getting closer to the boundary) is also referred to as scaffolding [28] or *curriculum design* [4,36,39]. The example sequences produced with this strategy are indeed favorable for certain learners [12,11,25].

There is also relevant work in the area of *intelligent tutoring systems*. For instance, work by Whitehill [45] models human learning as a partially observable Markov decision process, and defines the teaching problem as optimal control. Others have used algorithmic teaching to train crowd workers to correctly classify items such as butterfly species [35].

5.2. Interactive machine learning

A recently growing community within ML and Human–Computer Interaction investigates the interaction between humans and systems that involve ML. The term interactive ML was introduced by Fails and Olsen [13] to refer to the loop in which (i) the teacher provides new examples, (ii) the learner builds/updates the classifier, (iii) the learner gives feedback in the form of classifications on a large unlabeled dataset. The feedback from the learner affects the teacher's examples towards corrections of wrong classifications. Subsequent research in this area has extended interactive ML to a great number of systems, addressing a range of challenges from giving more control to users to providing them with useful information during interactions [34,14,22,16,8,31]. Our work shares the goal of this line of research, which is improving the effectiveness and usability of ML systems that interact with humans. However, our work differs in its focus on directly improving the teacher in this interaction, rather than improving the learner to deal with inefficiencies caused by the human teacher.

Some efforts in interactive ML are directed towards letting humans influence a classifier in ways other than presenting examples and labels. One example is the ManiMatrix system [18], that lets the user interact with the confusion matrix for a multi-class classifier to specify preferences about errors. It automatically reflects these preferences on the underlying classifier. Another example involves augmenting the learner's classifications with *explanations* and letting the user change weights of features of the classifier [38]. This has also been extended to *rule-based* and *similarity-based* explanations [37]. The authors analyze user critiques of the learner's explanation of a classification, and propose and evaluate several new ways that the user can directly influence the classifier, such as the ability to add different features (*e.g.* new words), or feature combinations (*e.g.* two consecutive words as one feature).

A different group of researchers focus on interactive agents in games that learn from human players [17,42,21,43]. These make observations about common patterns in human teaching (*e.g.* more positive feedback than negative, rewards diminishing over time) in order to augment the learner. Thomaz [41] introduced Socially Guided Machine Learning (SG-ML), which advocates designing learners to be more amenable to human teaching. Our work on teaching guidance is directly related to the aspect of SG-ML that calls for designing learning interactions that help the human teacher build the appropriate mental model of the learning process.

The problem of mental model soundness in interactions with learning systems was explicitly studied by Kulesza et al. [23]. They found that people who were given more training on how a recommendation system works, were able to train the system to better match their preferences. This work is highly complementary to our work—it demonstrates that giving people a more thorough understanding of the learner lets them provide better input, while our work demonstrates that directly telling people how to teach has the same effect. One of the key differences is that their study involved longer term interactions (*i.e.* 30 minutes training about the system and 5 days of interaction), as compared to our single session short interactions (10–20 minutes including everything). This difference intuitively indicates the contexts in which the alternative approaches would be preferable. In addition, we think that the complexity of the learner's operation might effect the choice between these two alternatives.

A few papers in the interactive ML literature share our goal of directly improving the input obtained from the teacher. Amershi et al. [1] investigate how the grouping and order in the presentation of large data sets to users, impact their input to a learner. Rosenthal and Dey [31] investigate how additional information provided to users when requesting labels, can reduce errors in their inputs. Our approach differs from these in the content and medium of information provided to users to improve their inputs. The information involves direct instructions on how to teach given to the teacher by a third party, rather than in the form of feedback from the learner itself.

6. Conclusions

Our paper demonstrates that humans do not naturally provide the best possible examples when they teach an arbitrary machine learner. We proposed *teaching guidance* as a mechanism to influence human teaching towards being more useful for the learner. We discussed how the nature of such guidance changes based on the problem type, and demonstrated its utility in three different domains. Across all of our experiments, teaching guidance helped people achieve better teaching performance.

References

[1] S. Amershi, J. Fogarty, A. Kapoor, D. Tan, Effective end-user interaction with machine learning, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2012, pp. 1529–1532.

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

[3] F. Balbach, T. Zeugmann, Recent developments in algorithmic teaching, in: Proceedings of the 3rd International Conference on Language and Automata Theory and Applications, 2009, pp. 1–18.

[4] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: L. Bottou, M. Littman (Eds.), Proceedings of the 26th International Conference on Machine Learning, Montreal, June 2009, pp. 41–48.

[5] M. Cakmak, M. Lopes, Algorithmic and human teaching of sequential decision tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), 2012, pp. 1536–1542.

[6] M. Cakmak, A. Thomaz, Mixed-initiative active learning, in: ICML 2011 Workshop on Combining Learning Strategies to Reduce Label Cost, 2011.

[7] H. Chernoff, The use of faces to represent points in k-dimensional space graphically, J. Am. Stat. Assoc. 68 (342) (1973) 361–368.

[8] A. Culotta, T. Kristjansson, A. McCallum, P. Viola, Corrective feedback and persistent learning for information extraction, Artif. Intell. 170 (14–15) (2006) 1101–1122.

[9] S. Dasgupta, Coarse sample complexity bounds for active learning, in: Advances in Neural Information Processing Systems (NIPS), 2006, pp. 235–242.
[10] F. Denis, R. Gilleron, PAC learning under helpful distributions, in: 8th International Workshop on Algorithmic Learning Theory, in: Lect. Notes Artif. Intell., vol. 1316, Springer-Verlag, 1997, pp. 132–145.
[11] O. Deniz, J. Lorenzo, M. Castrillon, J. Mendez, A. Falcon, Learning to recognize faces incrementally, in: Proceedings of the 29th DAGM Conference on Pattern Recognition, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 365–374.
[12] J. Elman, Learning and development in neural networks: the importance of starting small, Cognition 48 (1993) 71–99.
[13] J. Fails, D. Olsen, A design tool for camera-based interaction, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2003, pp. 449–456.
[14] J. Fogarty, D. Tan, A. Kapoor, S. Winder, Cueflik: interactive concept learning in image search, in: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems, 2008, pp. 29–38.
[15] S. Goldman, M. Kearns, On the complexity of teaching, J. Comput. Syst. Sci. 50 (1) (February 1995) 20–31.
[16] Y. Huang, T.M. Mitchell, Text clustering with extended user feedback, in: Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 413–420.
[17] C. Isbell, C. Shelton, M. Kearns, S. Singh, P. Stone, Cobot: a social reinforcement learning agent, in: Proceedings of the 5th International Conference on Autonomous Agents, 2001, pp. 377–384.
[18] A. Kapoor, B. Lee, D. Tan, E. Horvitz, Interactive optimization for steering machine classification, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2010, pp. 1343–1352.
[19] F. Khan, X. Zhu, B. Mutlu, How do humans teach: on curriculum learning and teaching dimension, in: Advances in Neural Information Processing Systems (NIPS), 2011.
[20] A. Kittur, E.H. Chi, B. Suh, Crowdsourcing user studies with mechanical turk, in: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems, 2008, pp. 453–456.
[21] W. Knox, P. Stone, Training a tetris agent via interactive shaping: a demonstration of the tamer framework, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2010, pp. 1767–1768.
[22] T. Kristjansson, A. Culotta, P. Viola, A. McCallum, Interactive information extraction with constrained conditional random fields, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2004, pp. 412–418.
[23] T. Kulesza, S. Stumpf, M. Burnett, I. Kwan, Tell me more? The effects of mental model soundness on personalizing an intelligent agent, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2012, pp. 1–10.
[24] C.H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 951–958.
[25] Y. Lee, K. Grauman, Learning the easy things first: self-paced visual category discovery, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 1721–1728.
[26] N. Littlestone, Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm, Mach. Learn. 2 (1988) 285–318.
[27] J. MacGregor, The effects of order on learning classifications by example: heuristics for finding the optimal order, Artif. Intell. 34 (3) (1988) 361–370.
[28] M. Mascolo, Change processes in development: the concept of coactive scaffolding, New Ideas Psychol. 23 (2005) 185–196.
[29] H.D. Mathias, A model of interactive teaching, J. Comput. Syst. Sci. 54 (3) (1997) 487–501.
[30] B. McCandliss, J. Fiez, A. Protopapas, M. Conway, J. McClelland, Success and failure in teaching the [r]-[l] contrast to Japanese adults: tests of a Hebbian model of plasticity and stabilization in spoken language perception, Cogn. Affect. Behav. Neurosci. 2 (2) (2002) 89–108.
[31] S. Rosenthal, A. Dey, Towards maximizing the accuracy of human-labeled sensor data, in: International Conference on Intelligent User Interfaces (IUI), 2010, pp. 259–268.
[32] J. Ross, L. Irani, M.S. Silberman, A. Zaldivar, B. Tomlinson, Who are the crowdworkers?: Shifting demographics in mechanical turk, in: Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems, 2010, pp. 2863–2872.
[33] P. Shafto, N. Goodman, Teaching games: statistical sampling assumptions for learning in pedagogical situations, in: Proceedings of the 30th Annual Conference of the Cognitive Science Society, 2008, pp. 1–6.
[34] M. Shilman, D. Tan, P. Simard, Cuetip: a mixed-initiative interface for correcting handwriting errors, in: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology, 2006, pp. 323–332.
[35] A. Singla, I. Bogunovic, G. Bartok, A. Karbasi, A. Krause, Near-optimally teaching the crowd to classify, arXiv preprint, arXiv:1402.2092, 2014.
[36] P. Stone, M. Veloso, Layered learning, in: R.L. de Mantaras, E. Plaza (Eds.), Proceedings of the Eleventh European Conference on Machine Learning, May/June 2000, Barcelona, Catalonia, Spain, Springer Verlag, 2000, pp. 369–381.
[37] S. Stumpf, V. Rajaram, L. Li, W. Wong, M. Burnett, T. Dietterich, E. Sullivan, J. Herlocker, Interacting meaningfully with machine learning systems: three experiments, Int. J. Hum.-Comput. Stud. 67 (8) (2009) 639–662.
[38] S. Stumpf, E. Sullivan, E. Fitzhenry, I. Oberst, W. Wong, M. Burnett, Integrating rich user feedback into intelligent user interfaces, in: Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI), 2008, pp. 50–59.
[39] M. Taylor, Assisting transfer-enabled machine learning algorithms: leveraging human knowledge for curriculum design, in: Proc. AAAI Spring Symposium on Agents that Learn from Human Teachers, 2009.
[40] J. Tenenbaum, V. Silva, J. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.
[41] A.L. Thomaz, Socially guided machine learning, Ph.D. thesis, Massachusetts Institute of Technology, 2006.
[42] A.L. Thomaz, C. Breazeal, Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance, in: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), 2006, pp. 1000–1005.
[43] L. VonAhn, L. Dabbish, Labeling images with a computer game, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2004, pp. 319–326.
[44] R. Warner, T. Stoess, P. Shafto, Reasoning in teaching and misleading situations, in: Proceedings of the 33rd Annual Conference of the Cognitive Science Society, 2011, pp. 1430–1435.
[45] J. Whitehill, A stochastic optimal control perspective on affect-sensitive teaching, Ph.D. thesis, University of California, San Diego, 2012.