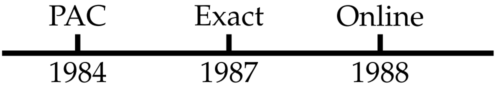
Notes on Computational Learning Theory and the problem of learning CNFs

March 5, 2018

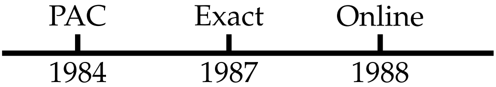
INTRODUCTION

- Computational Learning Theory lies between Machine Learning and Complexity Theory.



INTRODUCTION

- Computational Learning Theory lies between Machine Learning and Complexity Theory.



- Question posed in 1984: What is the complexity of learning Boolean Functions?
- Turing Award 2010: Leslie Valiant

PAC LEARNING: PROBLEM SETTING

- The learner observes a sequence of labeled examples (training).
- The learner must output a hypothesis estimating the target.
  - the hypothesis is evaluated by its performance on subsequent examples drawn according to a probability distribution.

PAC Learning: Motivation

- Given some training data over the general data, can we guarantee something about the error?
- How large should be the training data to bound the error?
- We want to compute a hypothesis where with high probability it does not differ much from the target.

PAC LEARNING: PROBLEM SETTING

- Let X be a set of examples.
- A concept c is a subset of X.
- A concept class $\mathcal{C}$ is a set of concepts.

PAC LEARNING: PROBLEM SETTING

- Let X be a set of examples.
- A concept c is a subset of X.
- A concept class $\mathcal{C}$ is a set of concepts.
- Let H be a set of hypothesis concept representations (the hypothesis space) and $\mu_H : H \to \mathcal{C}_H$ a surjective function;
- Let L be a set of target concept representations and $\mu_L : L \to \mathcal{C}_L$ a surjective function;

PAC LEARNING: PROBLEM SETTING

- Training examples are generated by a fixed, unknown probability distribution $\mathcal{D}$ over X (i.i.d).
  - $\mathcal{D} : X \to [0,1]$ is a function with $\sum_{x \in X} \mathcal{D}(x) = 1$.
- Let $\mathcal{D} = \{($$, 0.2), ($$, 0.1), ($$, 0.3), ($$, 0.2), ($$, 0.2)\}$ be a probability distribution over X.
- The oracle labels the examples as positive or negative according to the target.
  - e.g., if the target concept representation is { RGB() , RGB() } then:
    -  is a positive example;
    -  is a negative example.

TRUE ERROR OF A HYPOTHESIS

- The *true error* (denoted $error_{\mathcal{D}}(h)$) of a hypothesis $h \in H$ w.r.t. a target concept representation $l \in L$ and a prob. distr. $\mathcal{D}$ is the probability that h will misclassify an example drawn at random according to $\mathcal{D}$.

$$error_{\mathcal{D}}(h) = Pr_{x \sim \mathcal{D}}(x \in \mu_H(h) \oplus \mu_L(l))$$

True error of a Hypothesis

$$error_{\mathcal{D}}(h) = Pr_{x \sim \mathcal{D}}(x \in \mu_H(h) \oplus \mu_L(l))$$

Example

- Let $\mathcal{D}$ = {(, 0.2), (, 0.1), (, 0.3), (, 0.2), (, 0.2)} be a probability distribution over X.
- h = {RGB(), RGB()}
- l = { RGB() , RGB() }
- Then, $error_{\mathcal{D}}(h) = 0.3$

TWO NOTIONS OF ERROR

- **Training error** of a hypothesis h w.r.t. a target concept representation l.
  - How often $x \in \mu_H(h) \oplus \mu_L(l)$ over training examples?
- **True error** of a hypothesis h w.r.t. a target concept representation l.
  - How often $x \in \mu_H(h) \oplus \mu_L(l)$ over future random examples?
- Question: Can we bound the true error of h given the training error of h?

PAC Learning Definition

A learning framework $F = (X, L, H, \mu_H, \mu_L)$ is *PAC learnable* in polynomial time if there is an algorithm A such that for any fixed but arbitrary probability distribution $\mathcal{D}$ and any target $l \in L$:

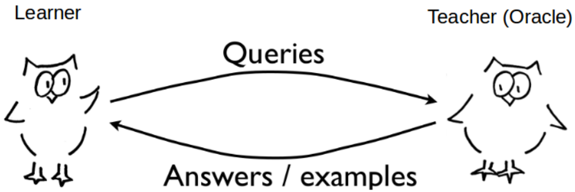
- A receives the parameters ϵ and δ as input;
- A can make calls to the oracle;
- the time used by A is bounded by a polynomial $p(|l|, |x|, \frac{1}{\epsilon}, \frac{1}{\delta})$, where $x \in X$ is the largest example returned by the oracle;
- A always halts and outputs a hypothesis $h \in H$ such that with probability at least $1 - \delta$, the probability of choosing $x \in \mu_H(h) \oplus \mu_L(l)$ is at most ϵ. That is, $Pr(Pr(x \in \mu_H(h) \oplus \mu_L(l)) \leq \epsilon) \geq 1 - \delta$.

PAC LEARNABILITY

- Question posed in 1984: What is the complexity of learning Boolean Formulas?
- The set of conjunctions of literals is PAC learnable in polynomial time from interpretations.
- 1987: Angluin proved that equivalence queries can be modified to achieve pac-learnability.
  - Exact Learning with membership and equivalence queries

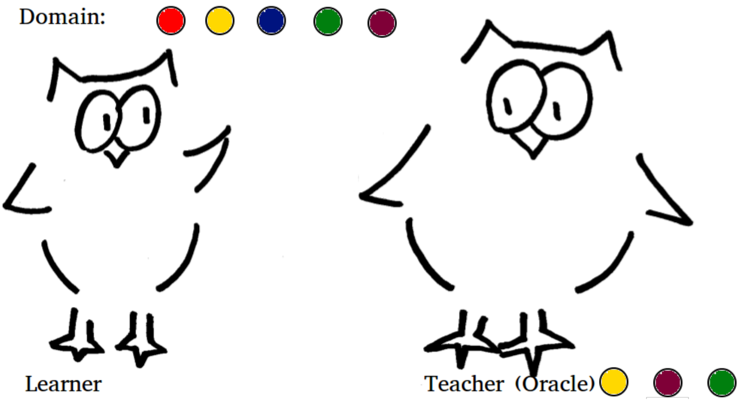
Angluin's Exact Learning Model

- An algorithm exactly identifies a target set L_* if it always halts and outputs a hypothesis L_h such that $L_h = L_*$.
  - Membership query: $x \in L_*$? Yes/No
  - Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



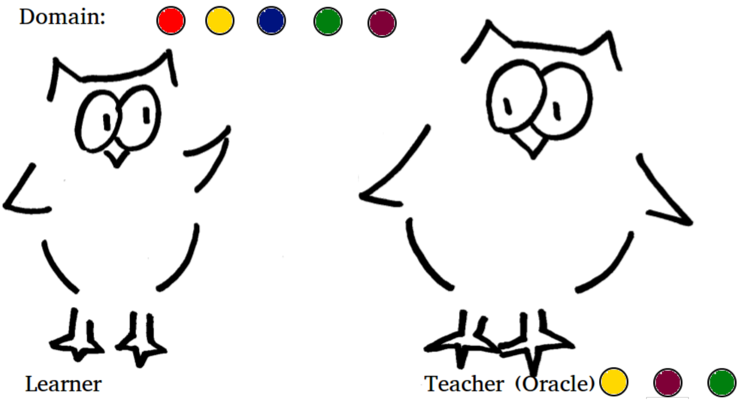
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



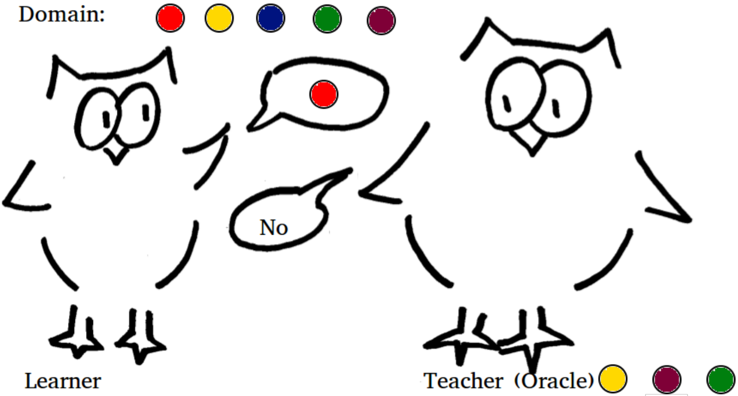
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



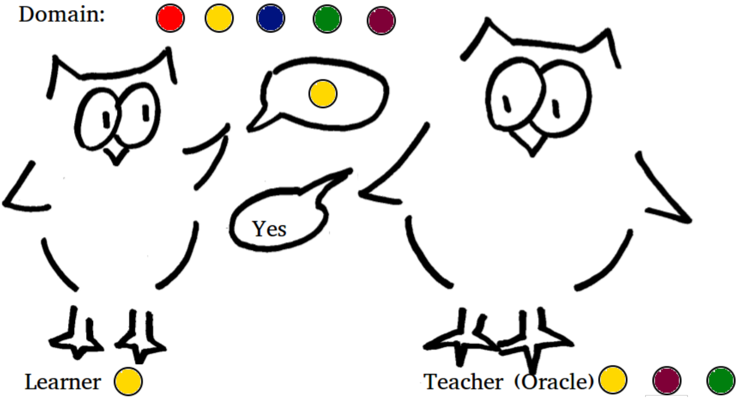
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



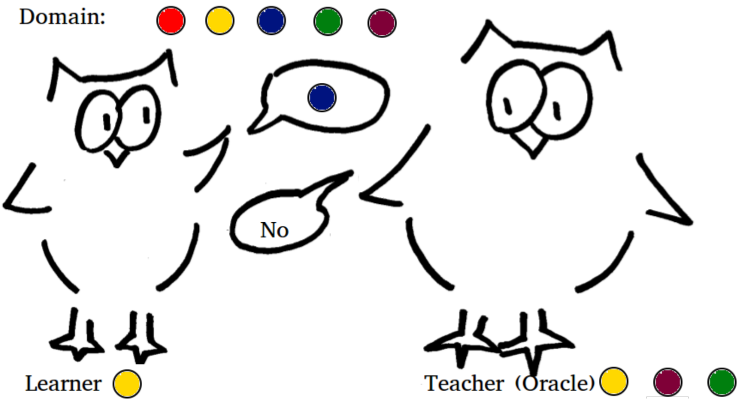
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



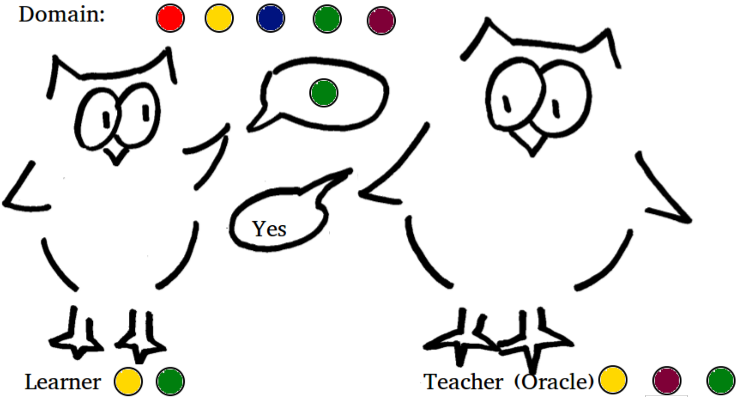
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



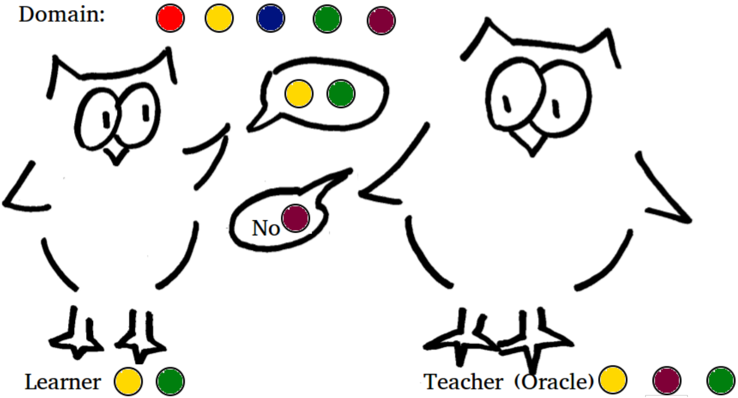
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



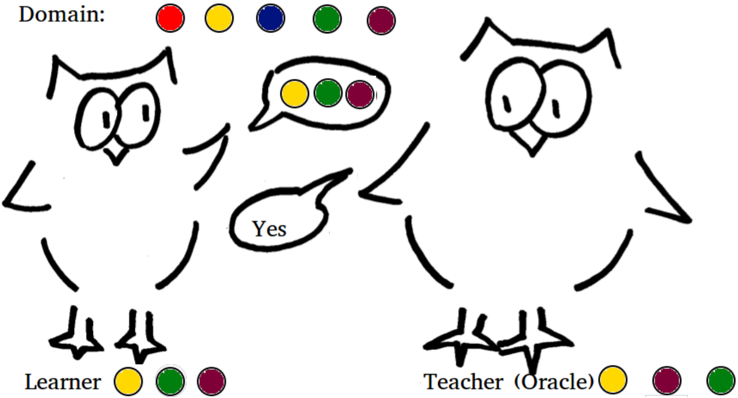
Exact Learning

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



EXACT LEARNING

- Membership query: $x \in L_*$? Yes/No
- Equivalence query: $L_h = L_*$? Yes/No and $x \in L_h \oplus L_*$



EXACT LEARNING

- Learnability in polynomial time:
  - Polynomial in the size of the target and the largest counterexample seen so far.
- Unknown:
  - CNFs (2-Quasi-Horn)

HORN $\prec$ MVDF $\prec$ 2-QUASI-HORN

- An MVDF is a set of (MVD) clauses $X \rightarrow Y \vee Z$.
  - $V = X \cup Y \cup Z$ and X, Y, Z are mutually disjoint.

Example:

- Propositional Variables: $\{a, b, c, d, e, f\}$
- Target MVDF:
  - $\{ab \rightarrow cd \vee ef, \quad c \rightarrow aef \vee bc, \quad abcd \rightarrow ef\}$
  - $\{\mathbf{T} \rightarrow abcd \vee ef, \quad abcdef \rightarrow \mathbf{F}\}$
- Each mvd clause $X \rightarrow Y \vee Z$ must contain all variables.

HORN $\prec$ MVDF $\prec$ 2-QUASI-HORN

Let $V = \{a, b, c, d, e, f\}$
Horn can be expressed as MVDF:

- Prop. Horn: at most one positive literal
  - $\{\neg a \vee \neg b \vee c\} \equiv \{ab \rightarrow c\}$
- Translation:
  - $\{ab \rightarrow c\} \equiv \{ab \rightarrow c \vee def, \quad abdef \rightarrow c\}$
  - Any interpretation of the form $(a, b, \neg c, ?, ?, ?)$ falsifies at least one of the two MVD clauses above.

HORN $\prec$ MVDF $\prec$ 2-QUASI-HORN

Let $V = \{a, b, c, d, e, f\}$
MVDF is a fragment of 2-Quasi-Horn:

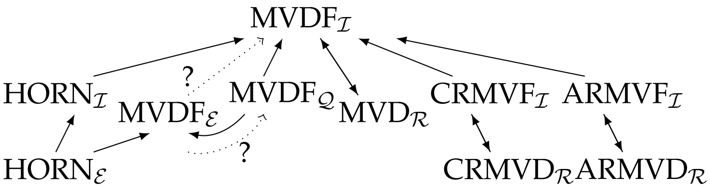
- 2-Quasi-Horn: at most two positive literals
  - $\{\neg a \vee b \vee c\} \equiv \{a \rightarrow b \vee c\}$
- Translation by distribution:
  - $\{ab \rightarrow cd \vee ef\} \equiv$
    $\{ab \rightarrow c \vee e, \quad ab \rightarrow c \vee f, \quad ab \rightarrow d \vee e, \quad ab \rightarrow d \vee f\}$

Our Exact Learning Problem

Horn	$\longrightarrow$	MVDF	$\longrightarrow$	2-Quasi-Horn
polytime learnable		?		as hard as CNF

- Horn-SAT: PTIME
- 2-Quasi-Horn-SAT: NP-Complete
- MVDF: PTIME - Exact Learning of Multivalued Dependency Formulas [Hermo and Ozaki, 2017]

OUR RESULTS



- HORN$_{\mathcal{I}}$ (1992): Angluin, Frazier and Pitt
- HORN$_{\mathcal{E}}$ (1993): Frazier and Pitt
- CRMVF$_{\mathcal{I}}$ and ARMVF$_{\mathcal{I}}$ (2011-2015): Lavin

Challenges of Learning MVDF

- The learning algorithm for propositional Horn refines countermodels by intersecting interpretations.
- In contrast to Horn, MVDF is not closed under intersection.
- Example: $\mathcal{T} = \{ab \rightarrow cd \vee ef\}$
  $\mathcal{I}_1 = (a, b, \neg c, \neg d, e, f)$ and $\mathcal{I}_2 = (a, b, c, d, \neg e, \neg f)$ satisfy $\mathcal{T}$
  but $\mathcal{I}_1 \cap \mathcal{I}_2 = (a, b, \neg c, \neg d, \neg e, \neg f)$ does not satisfy $\mathcal{T}$.

FUTURE WORK

- PAC-learning MVDF: q-bounded distributions
- Exact Learning: $\sqsubseteq_\Sigma$, $\equiv_\Sigma$

Hermo, M. and Ozaki, A. (2017).
Exact learning of multivalued dependency formulas.
Theoretical Computer Science.