Computational Learning Theory
Learning with Queries

Dimitris Diochnos
School of Computer Science
University of Oklahoma

Outline

1. Query Learning

Table of Contents

Learning Concepts using Queries

The material is based on Angluin's work [1].

- Want **complete identification** of a target concept $c \in \mathcal{C}$.
    - No probability distribution on instances.

Types of Queries

Membership: Ask if $x \in \mathcal{X}$ belongs to c.

- YES, if $x \in c$. Otherwise, NO.

Equivalence: Ask if $h = c$.

- YES if $h = c$; o/w NO and also return an $x' \in h \triangle c$.

Subset: Ask if $h \subseteq c$.

- YES if $h \subseteq c$; o/w NO and also return an $x' \in h \setminus c$.

Superset: Ask if $h \supseteq c$.

- YES if $h \supseteq c$; o/w NO and also return an $x' \in c \setminus h$.

Disjointness: Ask if $h \cap c = \emptyset$.

- YES if $h \cap c = \emptyset$; o/w NO and also return an $x' \in h \cap c$.

Exhaustiveness: Ask if $h \cup c = \mathcal{X}$.

- YES if $h \cup c = \mathcal{X}$; o/w NO and also return an $x' \notin h \cup c$.

- **Restricted queries** when the answer is YES/NO, but **no counterexample** is provided.

Learning Monotone Conjunctions

Membership Queries.

- $x_{ones} = (1, 1, 1, \ldots, 1, 1, 1)$ is a true (positive) point for every monotone conjunction.
- For $i \in \{1, \ldots, n\}$ ask all the instances x_i that differ from x_{ones} in position i only.

$$
\begin{array}{rcl}
x_{ones} & = & (1, 1, 1, \ldots, 1, 1, 1) \\
x_1 & = & (0, 1, 1, \ldots, 1, 1, 1) \\
x_2 & = & (1, 0, 1, \ldots, 1, 1, 1) \\
\vdots & & \vdots \\
x_{n-1} & = & (1, 1, 1, \ldots, 1, 0, 1) \\
x_n & = & (1, 1, 1, \ldots, 1, 1, 0)
\end{array}
$$

- If the answer is NO to x_i, then the i-th variable is relevant to the target.
- After exactly n membership queries we identify the target precisely.
  (the whole process runs in polynomial time $\Rightarrow$ it is efficient)

Learning Monotone Conjunctions

Equivalence Queries.

1. Initialize $h = x_1 \wedge x_2 \wedge \ldots \wedge x_n$.
2. If h is the target c, we are done.
3. Otherwise, we get a counterexample $x \in h \triangle c$. How does x look like?
   - h is satisfied only at $x_{ones} = (1, 1, 1, \ldots, 1, 1, 1)$ (x_{ones} satisfies *any* monotone conjunction)
   - Therefore, for any counterexample x that we will obtain:
     - x must be different from $x_{ones} \Rightarrow x$ contains at least one 0, and
     - x has to satisfy c.
   - As a consequence, the bits that are 0 in the counterexample x that we obtained, must correspond to variables that are not in the target c. $\Longrightarrow$ Update our guess for h by removing those variables from h for which we had a 0 in x. Then, go back to Step 2 and repeat similar arguments for the counterexamples obtained.

- The algorithm makes at most $n + 1$ queries.
  (the whole process runs in polynomial time $\Rightarrow$ it is efficient)

Learning General Conjunctions

Equivalence Queries.

1. First query for the identically false conjunction; e.g., $h_0 = x_1 \wedge \overline{x_1}$.
2. If h_0 is the target c, we are done.
3. Otherwise, we obtain a positive counterexample x.
   What can we infer from x?
   - x dictates the orientation of the variables in the target conjunction.
     - For example, if the i-th bit in x is a 0, then this implies that either variable x_i does not occur at all, or it occurs in the form of $\overline{x_i}$; it can not be the case that x_i appears as is in the target concept c.
4. From the previous step we have the starting hypothesis h_1, which is the full conjunction that satisfies the counterexample x that was returned. From this point and on we repeat the process that we followed earlier (for monotone conjunctions) until we determine the target concept precisely.

(the whole process runs in polynomial time $\Rightarrow$ it is efficient)

Learning General Conjunctions

Membership Queries.

- The previous algorithm does not generalize to general conjunctions.
- Reason: Lack of a starting truth assignment similar to x_{ones} that is positive for all the functions in the class.
- Adversary's Idea: As long as we have not asked all 2^n possible truth assignments, it is possible that:
  - c is satisfied by some unasked vector, or
  - c is identically false.

Negative Result. In order to learn an n-variable conjunction, one has to ask 2^n membership queries in the worst case.

- ... and this is true, regardless of the running time of the algorithm that updates the hypotheses ...

General Lower-Bound Techniques

Lemma 1 (Sunflower Lemma [1])

Suppose the hypothesis space $\mathcal{H}$ contains a class of distinct sets $L_1, \ldots, L_n$ and a set $L_\cap$ such that for any pair of distinct indices i and j we have: $L_i \cap L_j = L_\cap$. ($\mathcal{H}$ contains $n+1$ hypotheses.) Then, for any algorithm that exactly identifies each of the hypotheses L_i using restricted equivalence, membership, and subset queries, must make at least $n-1$ queries in the worst case.

– figure –

General Lower-Bound Techniques (cont'd)

Lemma 2 (Sunflower Lemma [1])

Suppose the hypothesis space $\mathcal{H}$ contains a class of distinct sets $L_1, \ldots, L_n$ and a set $L_\cap$ such that for any pair of distinct indices i and j we have: $L_i \cap L_j = L_\cap$. ($\mathcal{H}$ contains $n+1$ hypotheses.) Then, for any algorithm that exactly identifies each of the hypotheses L_i using restricted equivalence, membership, and subset queries, must make at least $n-1$ queries in the worst case.

Proof.

Restricted EQs. Response is NO for a query on L and at most one $L_i = L$ is eliminated.
Membership Queries. If $x \in L_\cap$, answer YES. Otherwise, answer NO $\Rightarrow$ remove at most one L_i.
Subset Queries. If $L \subseteq L_\cap$, answer YES. Otherwise, answer NO and any element in $L \setminus L_\cap$ is selected as the counterexample $\Rightarrow$ At most one L_i is eliminated. $\square$

General Lower-Bound Techniques (cont'd)

Theorem 3 (Double Sunflower [1])

Let $\mathcal{X} = \{x_1, \ldots, x_N, y_1, \ldots, y_N, z_1, z_2\}$; that is, $|\mathcal{X}| = 2N + 2$. Also, for $i \in \{1, \ldots, N\}$, let $c_i = \{z_1, x_i\} \cup \{y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_N\}$; that is, $|\mathcal{C}| = N$. Then, no matter which type of queries we use, we need $N - 1$ queries in order to identify any $c_i \in \mathcal{C}$.

Proof.

Equivalence Queries. When we ask c_j the answer is NO and x_j is returned as the counterexample. $\Rightarrow$ Removes at most 1 hypothesis ($h = c_j$)
Membership Queries. When we ask x_j, the answer is NO $\Rightarrow$ Removes at most 1 hypothesis ($h = c_j$)
When we ask y_j, the answer is yes $\Rightarrow$ Removes at most 1 hypothesis ($h = c_j$)
When we ask z_1, the answer is YES $\Rightarrow$ No removals
When we ask z_2 , the answer is NO $\Rightarrow$ No removals
Other Types of Queries. The paper discusses those in Section 6. $\square$

References

[1] Dana Angluin. Queries and Concept Learning. *Machine Learning*, 2(4):319–342, 1987.