Learning with Queries

These notes are slightly edited from scribe notes in previous years.

1 Equivalence Query (EQ)

We introduce the Equivalence Query. In this model, there are two actors: the learner and the adversary. The adversary is sometimes known as the world or the oracle. In the EQ, the learner presents a hypothesis, \( h(x) \) to the adversary. If the hypothesis is consistent with the adversary’s concept, then the adversary responds “yes”; otherwise the adversary responds “no,” and provides a counter example, \( CEX \). A CEX is an example \( x \) such that \( h(x) \neq \text{target}(x) \).

**Fact:** EQ can be simulated with random examples.

**How?** Given \( h \), take a sample of size \( m \geq \frac{1}{\epsilon} \log \frac{1}{\delta} \) examples. If \( h \) is consistent with all \( m \) examples, then say “yes,” otherwise provide one CEX that is inconsistent. Therefore, with probability \( > 1 - \delta \) we know that \( \text{error}(h) < \epsilon \).

**Definition:** \( C = \bigcup_{n \geq 1} C_n \) is learnable with EQ if there is a polynomial \( P() \) and an algorithm asking \( EQs \) such that \( \forall c \in C \) the algorithm asks \( \leq p(n, |c|) \) queries and stops with \( h \equiv C \). Hence the total run time is bounded by \( \text{poly}(n, |c|) \).

**Theorem:** if \( C \) is learnable with EQ, then it is PAC learnable.

The proof follows using a union bound with the previous fact.

2 Membership Query (MQ)

Here we introduce a membership query. Under membership queries, the learner invents a new data-point and asks the oracle for its label. (Machine invented examples may be problematic in practice since they might not correspond to “natural” examples in the domain and may be hard to classify.)

**Definition:** \( C = \bigcup_{n \geq 1} C_n \) is learnable with EQ and MQ if there is a polynomial \( P() \) and an algorithm asking EQ and MQ such that \( \forall c \in C \) the algorithm asks \( \leq p(n, |c|) \) queries overall and stops with \( h \equiv C \). (Like before, the total run time is bounded by \( \text{poly}(n, |c|) \)).
3 A detailed example: Monotone DNF

Definition: Monotone DNF is disjunctive normal form without negated literals. For example: 
\[ x_1 x_3 x_7 \lor x_1 x_2 \lor x_5 x_1 x_11. \]

The minimum representation of a monotone DNF is unique. Consider the monotone DNF 
\[ x_1 x_2 \lor x_3 = \text{target} \] shown in figure 2. In Algorithm 1 we show a method of learning the clauses in 
the target.

![Figure 2: A Lattice, showing the minimal representation of a monotone DNF. This example has 
eight nodes of three bits each. Those surrounded by circles are satisfying assignments to the 
monotone DNF. Each edge in the lattice indicates one bit flip, hence any adjacent nodes have 
Hamming distance = 1. The graph should be read from bottom up. Those nodes with dashed 
circles are the two minimal satisfying assignments, and they correspond to the clauses in the target 
concept. Those with solid circles are simply satisfying examples. Notice that the min points 
correspond to the terms of the DNF, for example 110 corresponds to \( x_1 x_2 \).]

Algorithm 1 Algorithm for Learning Monotone DNF

\[
S \leftarrow \{\} \text{ (Set of Min-points).} \\
\text{repeat} \\
\quad \text{EQR} \leftarrow \text{eq(terms}(S)) \\
\quad \text{if } \text{EQR} = \text{CEX then} \\
\quad \quad \text{// All counter examples are necessarily positive.} \\
\quad \quad \text{Walk down the lattice asking MQs with your CEX until you get a minpoint, } mp. \\
\quad \quad S \leftarrow S \lor mp \\
\quad \text{end if} \\
\text{until EQR = "true"}
\]

Lemma 1: if minpoint \( x \) is added to \( S \) then the term of \( x \) is in the target.
Corollary all counter examples are positive. That is, if \( \text{hyp}(x) = 1 \) then \( \text{target}(x) = 1 \).

Theorem: Algorithm 1 learns monotone DNF with \( \leq m + 1 \text{ EQ} \) and \( \leq mn^2 \text{ MQ} \). The latter can 
be imporved to \( mn \).

proof: Let \( m \) be the number of terms in the target. By Lemma 1, we know that we need only 
\( m + 1 \) equivalence queries. One for the empty set, and one of each conjunction we must learn. 
To account for membership queries, see Figure 3, and note that discovering each clause requires
tracking a counter example down from the top node (all ones). At each level we must query all of the child nodes. Hence each discovery requires at most $n^2$ MQs, and there are $m$ clauses to learn. (A slightly more careful analysis shows that $n$ queries suffice since we do not need to flip the same bit twice in this process.)

![Diagram: $n^2$ MQ Operations]

Figure 3: To discover each conjunction $n^2$ membership queries are needed.

4 Anti-Monotone CNF

Anti-Monotone CNF are Conjunctive Normal Form expressions with no positive literals. For example: $(\bar{x}_1 \lor \bar{x}_2) \land \bar{x}_3$. Here, the falsifying assignments are the same as those given in the lattice shown in Figure 2. Note the symmetry between the form of the example anti-monotone CNF, and the monotone DNF used in Figure 2. They can clearly be learned in a similar manner.

4.1 Horn CNF

Horn CNF are CNF that have at most one positive literal per clause. For example: $(\bar{x}_1 \lor \bar{x}_3 \lor x_5) \land (\bar{x}_1 \bar{x}_2) \land (\bar{x}_3 \lor x_4 \lor x_6)$ is a horn CNF. Using standard logical notation we can express the Horn clauses as rules $\bar{x}_1 \lor \bar{x}_3 \lor x_5 \equiv (x_1 \land x_3) \rightarrow x_5$. The part $(x_1 \land x_3)$ is called the condition or the antecedent whilst the literal after the “$\rightarrow$” is the conclusion or the consequence.

4.2 Violating a rule

A rule is violated if the condition is true, but the conclusion is false. In the example above, we violate the rule $(x_1 \land x_3) \rightarrow x_5$, with example “1 ⋆ 1 ⋆ 0 ⋆ ⋆” and satisfy it with “1 ⋆ 1 ⋆ 1 ⋆ ⋆.” Here “⋆” can be a binary value of your choosing.

4.3 Bitwise intersection

Definition: Bitwise intersection is computed over a two strings of bits, each of length $n$. For each bit, return 1 $\iff$ both corresponding bits in the input strings are 1. For example: 01101101 $\cap$ 11100000 = 01100000

4.4 An important result concerning Horn Expressions

A Theorem: A function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be expressed as a horn expression $\iff$ its set of positive examples is closed under bitwise intersection. That is, $\forall (x, y) \in \{0, 1\}^n, f(x) = f(y) = 1$
implies $f(x \cap y) = 1$.

Our proof covers only one direction of this result. We show that if $f$ is Horn and $f(x) = 1$ and $f(y) = 1$ then $f(x \cap y) = 1$. This is equivalent to saying that if $f(x \cap y) = 0$ then either $f(x) = 0$ or $f(y) = 0$.

If $f(x \cap y) = 0$ then there exists a rule in $f$ that is violated by $x \cap y$. A rule takes the form $C \rightarrow R$. Given that a rule is violated, $C$ is true in $x \cap y$ and $R$ is false in $x \cap y$. Since $C$ is true in $x \cap y$, $C$ must be true in $x$ and $C$ must be true in $y$ by the definition of bitwise intersection. But, $R$ is false in $x \cap y$, and $R$ must be false in either $x$ or $y$.