Outline
1. Warm Up
2. Knowledge Representations (disjunctions)
3. Algorithm for “learning” conjunctions.
4. PAC-Learning definition

1 Warm Up

Given a 5% chance that I will “give you up”, Pr[\(G\)], and a 10% chance that I will “let you down”, Pr[\(L\)], what is a bound probability that I do not give you up and let you down? In other words, what is the probability \(\neg\text{Pr}[G \land L]\)?

Answer:
\[
\text{Pr}[G \land L] = \text{Pr}[G] + \text{Pr}[L] - \text{Pr}[G \lor L]
\]
\[
\leq \text{Pr}[G] + \text{Pr}[L] \quad \text{(Since } \text{Pr}[G \lor L] > 0)\]
\[
\leq 15\%
\]
\[
\therefore \text{Pr}[G \land L] \geq 100\% - 15\% = 85\%.
\]
This approximation is called the Union Bound approximation. More formally:
\[
\text{Pr}[A_1 \land A_2 \land \ldots \land A_n] \leq \sum_{i=1}^{n} \text{Pr}[A_i]. \tag{1}
\]

2 Knowledge Representations

2.1 Disjunctions

Disjunctions are similar to conjunctions, but instead of being a series of literals joined with \(\land\), they are joined by logical OR operators, \(\lor\). Consider the following decision tree and corresponding truth table (Note that 1 and 0 correspond to true and false respectively):

![Decision Tree](https://example.com/decision-tree.png)

**Figure 1**: Decision Tree
Figure 2: Truth Table

Notice that the only conjunction that would be able to satisfy line 3 of the truth table in Figure 2 is $G \land \neg L$. Lines 1 and 3, however, clearly violate this conjunction.

It is possible to represent this function as a disjunction. From the decision tree in Figure 1, we see that the function is true when either $G$ is true OR $L$ is false. The corresponding disjunction, then, is:

$$G \lor \neg L$$

2.2 General Notes on Knowledge Representations

- Decision trees subsume both conjunctions and disjunctions.
- Decision trees are also able to represent logical functions that are not able to be represented by the union of conjunctions and disjunctions. (e.g., XOR or "parity" function)
- Algorithms that operate on decision trees are usually slower than algorithms for conjunctions.
- Generally, more expressive representations come at a cost.

3 Learning Conjunctions

The following objects are present in the PAC Learning Model:

- **Instance Space:** $X = \{0,1\}^n$ corresponds to the settings of $x_1, \ldots, x_n$.
- **Concept Class:** $C : X \mapsto \{0,1\}$, the collection of concepts (functions that map $X$ to $\{0,1\}$).
- **Target Concept:** $c \subseteq C$.

Examples: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ where $m$ is the number of examples.
Labels: $c(x^{(1)}) = b^{(1)}, c(x^{(2)}) = b^{(2)}, \ldots, c(x^{(m)}) = b^{(m)}$.

For learning conjunctions, $C$ is the class of all conjunctions of literals over $x_1, \ldots, x_n$. Our primary goal in a learning algorithm is to use a set of $m$ labeled examples to produce the representation of a concept $h$ ("hypothesis") that agrees with the unknown $c$.

Note:

- It’s hopeless to try to identify $c$ exactly.
- We can’t see all $x \subseteq X$.
- Even if we could, there might be multiple equivalent representations.
3.1 PAC Learning Algorithm for Conjunctions

In *Probably Approximately Correct (PAC) Learning*, we assume that examples are drawn *independently* from a common, unknown, arbitrary probability distribution $D$. Our goal in PAC Learning is to find a representation $h$ that approximates $c$ under $D$. The following Elimination Algorithm PAC learns the conjunction given by $c$.

**Algorithm 1** Elimination Algorithm

1: procedure Elimination($x = (x^{(1)}, x^{(2)}, \ldots, x^{(m)}), b = (b^{(1)}, b^{(2)}, \ldots, b^{(m)})$)
2: Initialize $h \leftarrow x^{(1)} \land x^{(2)} \land \ldots \land x^{(m)} \land \neg x^{(1)} \land \neg x^{(2)} \land \ldots \land \neg x^{(m)} = \bigwedge_{k=1}^{n} (x_k \land \neg x_k)$
3: for $i = 1$ to $m$ do
4: if $b^{(i)} = 1$ then
5: for $j = 1$ to $n$ do
6: if $x^{(j)} = 1$ then
7: remove $\neg x^{(j)}$ from $h$ if not already removed
8: end if
9: if $x^{(j)} = 0$ then
10: remove $x^{(j)}$ from $h$ if not already removed
11: end if
12: end for
13: end if
14: end for
15: return $h$
16: end procedure

- $h$ is initialized in line 2 to be the conjunction of all possible literals.
- The main idea is that the algorithm deletes any literals from $h$ that contradict the observed data.
- Runs in $O(mn)$ steps.

3.2 Proof of PAC-Learnability

**Theorem 1.** Given $m \geq (n/\epsilon)(\ln(2n) + \ln(1/\delta))$ examples, the Elimination Algorithm returns a representation $h$ such that with a probability of at least $1 - \delta$, $\Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon$.

**Note:**
In other words, for certain values of $\delta$ and $\epsilon$, the class of conjunctions is PAC-Learnable, and the Elimination Algorithm serves as a PAC-Learning Algorithm (more on this later).

**Proof.** A literal $l$ is only deleted from $h$ when $c(x) = 1$, but $l(x) = 0$. Then, $l(x)$ can’t be in $c(x)$. The only issue is that $h$ may contain literals that don’t appear in $c$.

Let’s say $\tilde{l}$ is “bad” if $\Pr_{x \in D}[\tilde{l} = 0 \lor c(x) = 1] > \epsilon/2n$.

**Claim:** If $h$ doesn’t contain any “bad” literals, then we can satisfy $h$ and satisfy the theorem’s condition.

$$\Pr_{x \in D}[c(x) \neq h(x)] \leq \sum_{\tilde{l} \in h} \Pr_{x \in D}[\tilde{l}(x) = 0 \land c(x) = 1].$$

(Union Bound)
If \( h \) contains no “bad” literals, 
\[
\Pr_{x \in D}[\tilde{l}(x) = 0 \land c(x) = 1] \leq 2n(\epsilon/(2n)) = \epsilon.
\]

Then, the theorem’s conditions are satisfied. For any “bad” literal \( \tilde{l} \), the probability that it is not removed from \( h \) after \( m \) examples is at most \((1 - \epsilon/(2n))^m\) for \( m \) independent examples.

Then, by another Union Bound over the 2\( n \) literals, the total probability that any “bad” literals remain in \( h \) is at most:
\[
\sum \text{“bad” } \tilde{l} (1 - \epsilon/(2n)) \leq 2n(1 - \epsilon/(2n))^m.
\]

For a specified number of examples, 
\[
m \geq ((2n)/\epsilon)(\ln(2n) + \ln(1/\delta)),
\]
we use the identity 
\[
(1 - x) \leq e^{-x}
\]
to find that the probability that \( h \) has a “bad” literal is at most:
\[
\sum \text{“bad” } \tilde{l} (1 - \epsilon/(2n))^m \leq 2ne^{-(\epsilon/(2n))m}
\]
\[
= 2n(\delta/(2n)) = \delta.
\]

\( \therefore h \) is adequate with probability greater than 1 - \( \delta \).

4 Definition of PAC-Learning

**Definition** Let \( C_n \) (for int \( n \)) be a class of real representations over \( x_n = x_1, x_2, \ldots, x_n \). Then, we say that the class, 
\[
C = \bigcup_{n=1}^{\infty} C_n,
\]
is **PAC-Learnable** if there exists an algorithm \( L \) that gives as parameters \( \epsilon \in (0,1/2) \) and \( \delta \in (0,1/2) \) and gives access to labeled examples of some concept \( c \in C_n \) which were drawn independently from a common distribution \( D \) over \( X \).

Additionally:

- \( L \) runs in polynomial time with respect to \( n, 1/\epsilon, 1/\delta, \) and \( \text{size}(c) \).
- With probability \( 1 - \delta \), \( L \) returns a representation \( h \) such that \( \Pr_{x \in D}[c(x) = h(x)] \geq 1 - \epsilon \).