| 6.858/18.428 Machine Learning | Lecture 1 : September 7, 1994 |
|---|---|
| *Lecturer: Ron Rivest* | *Scribe: Mona Singh* |

## 1.1 Introduction

What do we mean by "machine learning"? In this course, we will concentrate on *mathematical* models of machine learning. In particular, this course will focus on *computational machine learning* theory. There are two parts to computational learning theory:

- developing models (for example, PAC learning)
- deriving results within the model (for example, a typical result may show learnability or unlearnability within a model)

Both parts are important, but the first is particularly so, since ideally models are capable of capturing significant applications. In machine learning, there are many interesting and different models for learning. This is different from some other areas in computer science. For example, in computability theory, many different models were initially introduced, but were later shown to be equivalent, and thus the results within models are more important than the models themselves. However, in machine learning, different models give vastly different results for learnability and unlearnability.

## 1.2 Models of learning

There are several things which any "model of learning" must specify.

1. **Learner:** "Who" is doing the learning? Typically, in this course, we mean a computer program. The learner may be restricted. For example, often the learner is required to be efficient (polynomial time) or use only finite memory.

2. **Domain:** "What" is being learned? For example, the learner may be trying to learn an unknown concept, such as a chair. Concept learning, where the learner is trying to come up with a "rule" to separate positive examples from negative examples, is one of the most studied domains. There are many other types of things that can be learned: an unknown device (e.g., a programmable VCR), an unknown technique (e.g., how to juggle), an unknown function, an unknown environment (e.g., a new city), an unknown language, an unknown family of similar phenomena (e.g, speech recognition, face recognition, or character recognition), etc.

3. **Information Source:** "From what" is the learner learning? How is the learner informed about the domain? There are many ways this can happen:

   (a) **Examples:** The learner is given positive and negative examples. These examples can be chosen in a variety of ways. They can be chosen at random, from some known or unknown distribution. They can be chosen arbitrarily. They can be chosen maliciously, by some adversary who wants to know the worst-case behavior of a learning algorithm. Or, the examples can be carefully chosen by a helpful teacher who wants to facilitate the learning process.

   An important issue to address here is how are the examples described. That is, how are the features of the example specified?

   (b) **Queries:** The learner may get information about the domain by asking questions to a teacher. For example, the learner may ask "Is a stool an example of a chair?" Or, it may ask "Is this a correct floor plan for the third floor?" The first type of question is known as a *membership query*, where the learner asks for the label of an example. The second type of question is known as an *equivalence query*, where the learner provides a rule and asks if that rule is correct.

   (c) **Experimentation:** The learner may get information about the domain by actively experimenting with it. For example, a learner may learn how to program a VCR by playing with it. Or, it may learn a map of a new city by walking around it.

   An important issue is whether the model handles noisy or erroneous information sources. For example, a medical student may be trying to learn how to diagnose lung cancer from X-rays. She is given X-rays which a doctor has specified as either positive or negative examples of lung cancer. However, sometimes, a doctor may misread an X-ray. This results in some erroneous information. On the other hand, some of the X-ray images may be noisy.

4. **Prior Knowledge:** What does the learner know about the domain initially? This generally restricts the learner's uncertainty and/or biases and expectations about unknown domains. This tells what the learner knows about what is possible or probable in the domain. For example, the learner may know that the unknown concept is representable in a certain way. That is, the unknown concept might be known to be representable as a conjunction of features, or as a graph with at most 100 nodes. In practice, these types of assumptions can be very unrealistic. The learner may also know that a "simple" answer is preferable to a more "complex" answer. An important issue here is how to handle "incorrect" prior knowledge. How do you combine or trade-off prior vs. new information?

5. **Performance Criteria:** How do we know whether, or how well, the learner has learned? What is the learner learning for, and why? What is the learner's output? How does the learner demonstrate that it has learned something? For example, a video camera records everything around it, but we wouldn't say that it has learned anything. Difference performance criteria include:

   - Off-line (batch) vs. on-line (interactive) measures.
   - Descriptive output (e.g., representation of unknown concept) vs. predictive output (i.e., the learner need only label new instances as either positive or negative).
   - Accuracy. The learner is evaluated on the basis of its error rate, its correctness of description, or the number or mistakes it made during learning.
   - Efficiency. The learner may be evaluated on the amount of computation it does, and amount of information it needs (i.e., the number of examples it needs). The learner may be required to have asymptotic (or eventual) convergence, or it may be required to take polynomially bounded time.
   - Analysis assumptions. Sometimes, for analysis purposes, we assume things about the learning scenario. For example, we may assume the learner is given instances drawn from the uniform distribution. This might not actually be the case, but we sometimes assume this to make analysis possible. Note that this is different from the learner's prior knowledge.

## 1.3 On-line learning of conjunctive concepts with mistake-bound measure

As an example, let's consider the problem of on-line learning of boolean concepts. First, we will give the model, and then we will show that within this model it is possible to efficiently learn conjunctive concepts.

### 1.3.1 The model

- Learner: computer program
- Domain: Instance space $X = \{0,1\}^n$. That is, we have $n$-bit descriptors, $\overline{x} = v_1 v_2 \ldots v_n$, where the $v_i$'s are 0/1 variables. An unknown concept $c$ labels each example $\overline{x}$ as positive if $c(\overline{x}) = 1$ or as negative if $c(\overline{x}) = 0$.
- Source: A sequence of labeled examples, $< \overline{x}_1, c(\overline{x}_1) >, < \overline{x}_2, c(\overline{x}_2) >, \ldots$, which are picked arbitrarily and are noise-free.
- Prior knowledge: $c$ is a conjunction (e.g., $v_3 v_5 \overline{v}_8$).
- Performance: On-line prediction. That is, the learner must predict each $c(\overline{x})$ before it is seen. The performance measure is the number of prediction mistakes made.

### 1.3.2 Learning conjunctive concepts

The algorithm for learning conjunctive concepts works simply as follows. Let set $L$ contain the literals which the algorithm has not yet eliminated from being part of the conjunctive concept $c$ being learned. (A literal is a variable or its negation.) Initially, all literals are possibly in the conjunctive concept, and thus $L$ is the set of all literals. So, initially, $L = \{v_1, \overline{v}_1, v_2, \overline{v}_2, \ldots, v_n, \overline{v}_n\}$, and $|L| = 2n$. The learning algorithm always predicts by the conjunction of literals in $L$. Initially, the algorithm predicts 0 (false). Whenever a mistake is made, the learner removes from $L$ any literals that are false in the example.

**Theorem 1** *On-line learning of conjunctive concepts can be done with at most $n+1$ prediction mistakes.*

**Sketch of Proof:** We prove this theorem by the following lemmas. Lemma 1 shows correctness. Note that the worst-case occurs when the concept $c$ to be learned is **true**. That is, all literals must be eliminated from $L$. Lemmas 3 and 4 imply that at most $n+1$ mistakes are made. ■

**Lemma 1** *No literal in the concept $c$ to be learned is ever removed from $L$.*

**Proof:** Suppose not. Then, consider the first literal to be removed from $L$ which is also in the concept $c$. Before this literal is removed, all literals of $c$ are in $L$, and thus any example which satisfies all literals in $L$ also satisfies $c$. Thus the first such removal must have happened on a positive example for $c$ which does not satisfy some literal in $L$. But, a positive example for $c$ satisfies all its literals, and thus none of the literals of $c$ are removed. Contradiction. ■

**Lemma 2** *Mistakes are only made on positive examples.*

**Proof:** The algorithm always predicts by the conjunction of all literals in $L$. The concept $c'$ consisting of the conjunction of all literals in $L$ is the most specific concept consistent with all examples seen so far, since removing any of its literals gives another concept which has as positive examples all the positive examples of $c'$. (For example, concept $x_1x_3x_7$ is more specific than concept $x_1x_7$. That is, any positive example for $x_1x_3x_7$ satisfies $x_1$ and $x_7$, and thus also is a positive example for $x_1x_7$. Note, however, that positive examples for $x_1x_7$ might not be positive examples for $x_1x_3x_7$. On the other hand, a negative example for concept $x_1x_7$ doesn't satisfy at least one of $x_1$ and $x_7$, and thus is a negative example for $x_1x_3x_7$ as well. See figure 1.1.) So, all negative examples for the concept $c$ are predicted to be negative by this method, and thus mistakes are only made on positive examples. ■

**Lemma 3** *Each mistake causes $\geq 1$ literal to be removed from $L$.*

**Proof:** Assume that after some mistake, no literals are removed. This implies that the example for which a mistake was made satisfies all literals in $L$. Thus, by the prediction scheme specified above, the algorithm predicted positive for this example, and since it made a mistake, this example must be negative. However, by Lemma 2, mistakes are only made on positive examples. Contradiction. ■

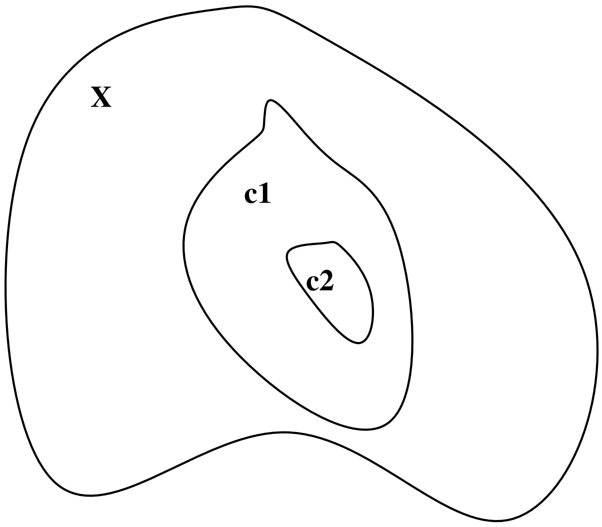


Figure 1.1: The set $X$ is the set of all instances. Concept $c2$ is more specific than concept $c1$. That is, the positive instances of $c2$ are also positive instances of $c1$, and all the negative instances of $c1$ are also negative instances of $c2$.

**Lemma 4** *The first mistake causes $n$ literals to be removed from $L$.*

**Proof:** Each example has exactly $n$ literals true, and the other $n$ literals false. When the first mistake is made all $n$ literals which are false can be removed from $L$. ■