Machine learning theory

Consistency model

Hamid Beigy

Sharif University of Technology

February 14, 2022

Table of contents

Introduction

1. To study machine learning mathematically, we need to formally define the learning problem.
2. This precise definition is called a learning model.
3. A learning model should be rich enough to capture important aspects of real learning problems, but simple enough to study the problem mathematically.
4. As with any mathematical model, simplifying assumptions are unavoidable.
5. A learning model should answer several important questions:
   - What is it that we are trying to learn?
   - What kind of data is available to the learner?
   - In what way is the data presented to the learner (online, actively, etc.)?
   - What type of feedback does the learner receive, if any?
   - What is the learner's goal?
6. A good learning model should also be robust to minor variations in its definition.

Definition

1. Let Σ be a set called alphabet for describing examples and assume that $\Sigma = \{0, 1\}$) or $\Sigma = \mathbb{R}$.
2. Let Σ^n be the set of n-tuples of Σ.
3. Let Σ^* be the set of non-empty finite strings of elements of Σ.

Definition (Domain set)

The set $\mathcal{X} \subseteq \Sigma^*$ is called domain set or example space and its members as examples.

Definition (Concept)

A concept over the alphabet Σ is a function $c : \mathcal{X} \mapsto \{0, 1\}$ and the set $\mathcal{C} = \{c \mid c : \mathcal{X} \mapsto \{0, 1\}\}$ with its associated representation is called concept class.

Definition (Training set)

A set/sequence $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is called training set or training examples.

Definition (Positive/negative examples)

1. An example $x \in \mathcal{X}$ for which $c(x) = 1$ is known as a positive example.
2. An example $x \in \mathcal{X}$ for which $c(x) = 0$ is known as a negative example.

Example (Parity concept)

Let $\Sigma = \{0, 1\}$ and define $p : \Sigma^* \mapsto \{0, 1\}$ as follows: if $x = x_1 x_2 \dots x_n$, then

$$p(x) = \begin{cases} 1 & \text{if an odd number of } x_i\text{'s are } 1 \\ 0 & \text{otherwise.} \end{cases}$$

This concept is known as the parity concept.

- The string 1101010 is a negative example.
- The string 11101010 is a positive example.

Example (Unit ball)

Let $\Sigma = \mathbb{R}$ and define $u : \Sigma^n \mapsto \{0, 1\}$ as follows:

$$u(x_1 x_2 \dots x_n) = \begin{cases} 1 & \text{if } x_1^2 + \dots + x_n^2 \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

This concept is known as the n-dimensional unit ball.

Definition (Formal mode learning)

1. Learner's input: the learner has access to the following:
   - Domain set $\mathcal{X}$
   - Label set $\mathcal{Y}$, we assume that $\mathcal{Y} = \{0, 1\}$.
   - Training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$.
2. Learner's output: the learner a hypothesis $h : \mathcal{X} \mapsto \mathcal{Y}$.
3. Data generation model $\mathcal{D}$. We assume that each $x \in \mathcal{X}$ is sampled according distribution $\mathcal{D}$, which is unknown to the learning algorithm.
4. Measures of success:

   Training error

$$\hat{\mathbf{R}}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{I}\left[h(x_i) \neq c(x_i)\right]$$

   True error

$$\mathbf{R}(h) = \mathop{\mathbb{P}}_{x \sim \mathcal{D}} \left[h(x) \neq c(x)\right]$$

5. Information available to the learner

Consistency model

The consistency model is not a particularly great model of learning, but it's simple and good to start.

Definition (Consistency model)

We say that algorithm $\mathcal{A}$ learns the concept class $\mathcal{C}$ in the consistency model if given any training set S, the algorithm produces a hypothesis (concept) $c \in \mathcal{C}$ consistent with S **if one exists**, and outputs **"there is no consistent concept"** otherwise.

Definition (Learnability of consistency model)

We say that a class $\mathcal{C}$ is **learnable** in the consistency model **if there exists an efficient algorithm** $\mathcal{A}$ that learns $\mathcal{C}$ in the consistency model.

Here **efficient** means that **the algorithm runs in polynomial time** in terms of the size of the set S and the size of each $x \in S$.

Example (Learnability of monotone conjunctions)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all monotone conjunctions (AND of a subset of the (unnegated) variables, such as $c(x) = x_2 \wedge \bar{x}_7 \wedge x_9$.
3. A sample training set is given in the following table

01101	+
11011	+
11001	+
00101	-
11000	-

4. Is any learning algorithm for learning this concept class??
5. **we can learn this class in the consistency model by taking the bitwise AND of all of the positive examples, then form a conjunction of all variables corresponding to the bits that are still on.**
6. For the above training set, we obtain $c(x) = x_2 \wedge x_5$.
7. Is this hypothesis consistent with the negative examples?
8. Is this algorithm efficient?

Example (Learnability of conjunctions)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all conjunctions (AND of a subset of the (possibly negated) variables, such as $c(x) = x_2 \wedge x_7 \wedge x_9$.
3. Is any learning algorithm for learning this concept class??
4. The best way is **to reduce the problem of learning conjunctions to the problem of learning monotone conjunctions and just use our previous algorithm** such as
   - First, for each variable x_i, introduce a new variable $z_i = \bar{x}_i$ representing its negation.
   - Then, we extend each n-bit example to $2n$-bit by concatenating each example with its negation.

   Initial training set

01101	+
11101	+
11100	+
01111	-
11000	-

   Extended training set

0110110010	+
1110100010	+
1110000011	+
0111110000	-
1100000111	-

   - Finally, applying the monotone conjunction learning algorithm.
5. For the above training set, we obtain $c(x) = x_2 \wedge x_3 \wedge z_4 = x_2 \wedge x_3 \wedge \bar{x}_4$.
6. Is this hypothesis consistent with the negative examples?
7. Does consistent monotone conjunction exist for these examples.
8. Is this algorithm efficient?

Example (Learnability of monotone disjunctions)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all monotone disjunctions (OR of a subset of the unnegated) variables, such as $c(x) = x_2 \vee x_7 \vee x_9$.
3. Is any learning algorithm for learning this concept class?
4. The best way is **to reduce the problem of learning monotone disjunctions to the problem of learning monotone conjunctions and just use our previous algorithm**.
5. We can use DeMorgan's Law from logic

$$\boxed{(x_1 \vee \ldots \vee x_n) = \overline{(\bar{x}_1 \wedge \ldots \wedge \bar{x}_n)}}$$

6. and educe the monotone disjunction problem to the monotone conjunction problem by
   - Flipping all of the bits in the training set.
   - Flipping all labels in the training set.
   - Applying the monotone conjunction algorithm and finding a concept c.
   - Negating all of the literals in c and then negating the conjunction itself.
7. For the given training set, we obtain $c(x) = x_2 \wedge x_3 \wedge z_4 = x_2 \wedge x_3 \wedge \bar{x}_4$.
8. Is this hypothesis consistent with the negative examples?
9. Is this algorithm correct? (prove it.)
10. Is this algorithm efficient?

Example (Learnability of k-CNF formulas)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all k-CNF formulas that is conjunctions of disjunctions (called clauses) where each disjunction has at most k literals.
3. For example, for $k = 2$, we have $c(x) = (x_2 \vee x_7) \wedge (x_{11}) \vee (x_4 \vee \bar{x}_9)$.
4. Is any learning algorithm for learning this concept class?
5. The best way is **to reduce this problem to the problem of learning monotone conjunctions and just use our previous algorithm**.
   - First, creating a new variable for every possible clause (disjunction) that could appear in our k-CNF formula.
   - For example, for $n = k = 2$, we create

$$z_1 = (x_1) \quad z_2 = (x_2)$$

$$z_3 = (x_1 \vee x_2) \quad z_4 = (x_1 \vee \bar{x}_2)$$

$$z_5 = (\bar{x}_1 \vee x_2) \quad z_6 = (\bar{x}_1 \vee \bar{x}_2)$$

   - We converted the two-bit examples into six-bit examples.
   - Then, applying monotone conjunction algorithm resulting in a conjunction c of z_i's.
   - Finally, converting these z_i's into their corresponding disjunctions.
6. Is this algorithm consistent? (prove it.)
7. Is this algorithm correct? (prove it.)
8. Is this algorithm efficient?

Lemma (Learnability of k-CNF formulas)

The algorithm designed for learning k-CNF formulas is not efficient.

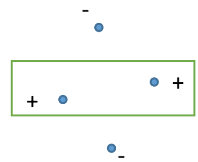
Proof.

1. The number of z_i variables equals to $O\left((2n)^k\right)$.
2. The number of k-CNF's with n variables equals to $(2n)(2n-1)\dots(2n-k) = O\left((2n)^k\right)$.
3. This is because each position in the k-CNF has $2n$ possible choices of variables, all the of x_i and their negations.
4. The algorithm is thus **efficient** (polynomial time) if we assume k to be a small constant but **not otherwise**.

$\square$

Example (Learnability of axis-aligned rectangles)

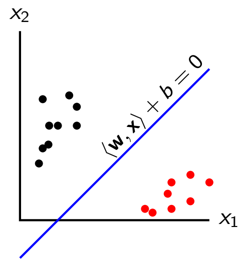
1. Let $\mathcal{X} = \mathbb{R}^2$ be the set of all points in two-dimensional space.
2. Let the concept class $\mathcal{C}$ consist of all 2-D axis-aligned rectangles such as



3. We can find a rectangle containing all positive examples without containing any negative examples using the following algorithm.
   - Finding $x_1^{min} = min\{x_{11}, \ldots, x_{1m}\}$.
   - Finding $x_1^{max} = max\{x_{11}, \ldots, x_{1m}\}$.
   - Finding $x_2^{min} = min\{x_{21}, \ldots, x_{2m}\}$.
   - Finding $x_2^{max} = max\{x_{21}, \ldots, x_{2m}\}$.
   - Then c is the rectangle defined by points $\left(x_1^{min}, x_2^{min}\right)$ and $\left(x_1^{max}, x_2^{max}\right)$
4. Show that this rectangle is the smallest rectangle that can possibly contain all the positive examples and none negative examples.
5. Is this algorithm consistent? (prove it.)
6. Is this algorithm correct? (prove it.)
7. Is this algorithm efficient?

Example (Learnability of half hyperspaces)

1. Let $\mathcal{X} = \mathbb{R}^n$ be the set of all points $x = (x_1, \ldots, x_n)$ in n-dimensional space.
2. Let the concept class $\mathcal{C}$ consist of all half hyperspaces (linear threshold functions) such as



3. In other words, we want a weight vector $\mathbf{w}$ and some threshold b such that
   - $\langle \mathbf{w}, \mathbf{x}_i \rangle > b$ if $y_i = 1$.
   - $\langle \mathbf{w}, \mathbf{x}_i \rangle < b$ if $y_i = 0$.
4. Since the $\mathbf{x}_i$ are all known, this results in a **simple linear program** (will be given later).
5. Is this algorithm consistent? (prove it.)
6. Is this algorithm correct? (prove it.)
7. Is this algorithm efficient?

Example (Learnability of 2-term DNF)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all 2-term DNF that is OR of two arbitrary length conjunctions.
3. For example, we have $c(x) = (x_2 \wedge x_7 \wedge x_8) \vee (x_4 \wedge \bar{x}_9)$.
4. Is any learning algorithm for learning this concept class?
5. **Can we reduce this problem to the problem of finding a k-CNF?**
6. Informally, disjunction can be treated as multiplication and conjunction can be treated as addition.
7. As an example, $(x_1 \wedge x_2) \vee (x_3 \wedge x_4) = (x_1 \vee x_3) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_3) \wedge (x_2 \vee x_3)$.
8. This implies that we can always convert 2-term DNF's into 2-CNF's, but does this mean the class is learnable?
9. We can always run our 2-CNF learning algorithm and find a consistent 2-CNF and convert this 2-CNF into a 2-term DNF (if possible).
10. **But it is not possible always, because** $(2 - term\ DNF) \subseteq (2 - CNF)$.
11. Is this algorithm efficient? **From the fact that learning 2-CNF's is easy, learning 2-term DNF's is NP-hard.**
12. Here we arrive at a fundamental problem with our consistency model. **It is possible for a class $\mathcal{C}$ to be learnable but to have a subclass of $\mathcal{C}$ be unlearnable.**

Example (Learnability of DNF)

1. Let $\mathcal{X} = \{0, 1\}$ be the set of all n-bit vectors.
2. Let the concept class $\mathcal{C}$ consist of all DNF that is the OR of an arbitrary number of arbitrary-length conjunctions.
3. For example, we have $c(x) = (x_2 \wedge x_7 \wedge x_8) \vee (x_4 \wedge \bar{x}_9) \vee (x_3 \wedge \bar{x}_5 \wedge x_7)$.
4. Is any learning algorithm for learning this concept class?
5. **Construct a clause for each positive example with a literal corresponding to the truth value of each bit.**

01101	+	$(\bar{x}_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5)\vee$
11101	+	$(x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge x_5)\vee$
11100	+	$(x_1 \wedge x_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5)$
01111	-	
11000	-	

6. This method is effective and efficient at learning a DNF that is consistent with the input.
7. But what is this DNF even useful for?
8. This example highlights a distinction between **memorization** and **generalization.**

Summary

Summary

The examples showed a few shortcomings of the consistency model.

1. A class $\mathcal{C}$ can be learnable while a subclass of $\mathcal{C}$ can be unlearnable.
2. The consistency model yields a concept that tells us nothing about the accuracy of the model on new data (generalization).
3. The consistency model has a practical problem in that training data that contains noise is not handled in a robust way.

Questions?