"Sapienza" University of Rome
Faculty of Information Engineering, Informatics and Statistics
Department of Computer Science

Machine Learning

Lecture notes integrated with the book TODO

Author
Alessio Bandiera

October 5, 2024

Contents

Information and Contacts

Personal notes and summaries collected as part of the *Machine Learning* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/aflaag-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **alessio.bandiera02@gmail.com**

- LinkedIn: **Alessio Bandiera**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

TODO

Licence:

These documents are distributed under the **GNU Free Documentation License**, a form of copyleft intended for use on a manual, textbook or other documents. Material licensed under the current version of the license can be used for any purpose, as long as the use meets certain conditions:

- All previous authors of the work must be **attributed**.

- All changes to the work must be **logged**.

- All derivative works must be **licensed under the same license**.

- The full text of the license, unmodified invariant sections as defined by the author if any, and any other added warranty disclaimers (such as a general disclaimer alerting readers that the document may not be accurate for example) and copyright notices from previous versions must be maintained.

- Technical measures such as DRM may not be used to control or obstruct distribution or editing of the document.

1

TODO

1.1 Learning problems

1.1.1 Learning

A **learning problem** is defined by the following *three components*.

> **Definition 1.1: Learning**
>
> **Learning** is defined as *improving*, through *experience E*, at some *task T*, with respect to a *performance measure P*.

Example 1.1 (Learning problems)**.** Consider the problem of learning how to play Checkers; in this example, the *task T* is to be able to play the game itself, the *performance measure P* could be the percentage of games won in a tournament, but *experience E* is more complex.

In general, *experience* can be acquired in several ways:

- in this example, a human expert may suggest optimal moves for each configuration of the board; however, this approach may not generalize for any problem, as human experts may not exists for certain tasks;

- alternatively, the computer may play against a human, and automatically detect win, draw and loss configurations;

- lastly, the computer may play against itself, learning from its own successes and failures.

For this particular game, a possible **target function** (the function that would be useful to learn in order to solve the learning problem) could be the following

$$\text{ChooseMove} : \text{Board} \rightarrow \text{Move}$$

which, given a board state, returns the best move to perform, but also

$$V : \text{Board} \to \mathbb{R}$$

which assigns a *score* to a given board.

For instance, consider the following target function:

$$V(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

where b is a given *board state*, and

- $bp(b)$ is the number of *black pieces*
- $rp(b)$ is the number of *red pieces*
- $bk(b)$ is the number of *black kings*
- $rk(b)$ is the number of *red kings*
- $bt(b)$ is the number of *red pieces threatened by black pieces*
- $rt(b)$ is the number of *black pieces threatened by red pieces*

In this formulation, V is a *linear compbination* of multiple coefficients w_i, which are unknown. Therefore, in this example **goal** of the *learning problem* is to **learn** V, or equivalently, to **estimate each coefficient** w_i. Note that this function *can be computed.*

1.1.2 Machine Learning problems

> **Definition 1.2: Dataset**
>
> Let $V(b)$ be the *true target function* (always *unknown*), $\hat{V}(b)$ be the *learned function* — an approximation of $V(b)$ computed by the *learning algorithm* — and $V_t(b)$ the *training value* of b in the *training data*. Lastly, let X be an input domain.
>
> Given a set of n inputs
> $$X_D := \{b_i \mid i \in [1, n]\} \subset X$$
> a **dataset** is a set of *samples*, and it is denoted as
>
> $$D = \{(b_i, V_t(b_i)) \mid b_i \in X_D\}$$

In the previous example, $\hat{V}(b)$ would have the following form

$$\hat{V}(b) = \hat{w}_0 + \hat{w}_1 \cdot bp(b) + \hat{w}_2 \cdot rp(b) + \hat{w}_3 \cdot bk(b) + \hat{w}_4 \cdot rk(b) + \hat{w}_5 \cdot bt(b) + \hat{w}_6 \cdot rt(b)$$

> **Definition 1.3: Machine Learning problem**
>
> A **machine learning problem** is the *task* of *learning a function* $f : X \to Y$, given a *dataset* D.
>
> To **learn a function** f means *computing an approximation function* $\hat{f}$ that returns values as close as possible to f, especialy for values *outside* D
>
> $$\forall x \in X - X_D \quad \hat{f}(x) \approx f(x)$$

Note that $|X_D| << |X|$, which makes the task of learning f quite challenging.

There are multiple types of Machine Learning (ML) problems, such as *dataset type* and *target function type*. The various ML problems will be discussed in later sections.

1.1.3 Hypotheses

> **Definition 1.4: Hypothesis space**
>
> Given an ML problem, an **hypothesis** h for the problem is an approximation of its target function, and its **hypothesis speace** H is the set of all possible hypothesis, i.e. the set of all functions that can be learned, which correspond to all the approximations of the target function of the ML problem.

Given this definition, **learning** can be defined as *searching in the hypothesis space*, using the dataset D and some performance function P of the given ML problem

$$h^* \in \arg\max_{h \in H} P(h, D)$$

A **performance measure** is a metric that evaluates the correctness of a given hypothesis, by comparing $h(x)$ and $f(x)$ for all $x \in X_D$, where f is the target function of the ML problem.

Example 1.2 (Hypothesis). Consider the ML problem of *classifying natural numbers into primes and composite numbers*. The *target function* would be the following

$$f : \mathbb{N} \to \{\mathbb{P}, \mathbb{N} - \mathbb{P}\}$$

A dataset D for this ML problem would look like the following example

$$D = \{(1, \mathbb{P}), (3, \mathbb{P}), (5, \mathbb{P}), (6, \mathbb{N} - \mathbb{P}), (8, \mathbb{N} - \mathbb{P}), (10, \mathbb{N} - \mathbb{P})\}$$

The hypothesis space is the set of all possible *classification functions* of the form

$$h_A : \mathbb{N} \to \{A, \mathbb{N} - A\}$$

placeholder

repr vs gener??

Definition 1.5: Hypothesis consistency

Given an ML problem defined by a target function $c : X \to Y$ — for some sets X and Y — and a training dataset $D = \{(x, c(x))\}$, an hypothesis $h \in H$ is said to be **consistent with** D if and only if

$$\forall x \in D \quad h(x) = c(x)$$

Note that this definition is important, because $h(x)$ can be evaluated for any $x \in X$, but only inputs that appear in the dataset can be verified, for which $c(x)$ is known. Therefore, *consistency* should be desirable for an hypothesis, since the real goal of an ML system is to find *the best h* that predicts correct values of $h(x')$, for instances $x' \notin X_D$, with respect to the unknown values $c(x')$.

Definition 1.6: Inductive learning hypothesis

The **inductive learning hypothesis** states the following:

> Given an ML problem, any hypothesis that approximates the target function well over a sufficiently large set of training examples, will also approximate the target function well over other unobserved examples.

placeholder ── paragraph?

Definition 1.7: Version space

The **version space** $VS_{H,D}$ of an ML problem, is the subset of hypotheses of H consistent with all training examples in D. Using symbols

$$VS_{H,D} := \{h \in H \mid \forall x \in X_D \quad h(x) = c(x)\} \subset H$$

Algorithm 1.1: List-Then-Eliminate

Given an ML problem, the algorithm returns $VS_{H,D}$.

1: **function** LISTTHENELIMINATE(X_D, D)
2:     $VS_{H,D} := H$ ▷ initially it contains any hypothesis
3:     **for** $(x, c(x)) \in D$ **do**
4:         $H' := \{h \in H \mid h(x) \neq c(x)\}$ ▷ set of inconsistent hypotheses for x
5:         $VS_{H,D} = VS_{H,D} - H'$
6:     **end for**
7:     **return** $VS_{H,D}$
8: **end function**

This algorithm can theoretically find the version space for any ML problem, but *it is not*

computable, as it requires to **enumerate all the possible hypotheses**.

1.1.4 Representation issues

Consider a *binary classification* ML problem — commonly referred to as **Concept Learning** (CL) — , and its hypotheses space H; usually, every hypothesis is associated to the set of the instances that are classified as 1 by such hypothesis

$$\phi : H \to \mathcal{P}(S) : h \mapsto \{x \in X \mid h(x) = 1\}$$

note that it is not always true that, for any set $S \subseteq X$, there exists an h such that for each $x \in S$, $h(x) = 1$. Assume that, for the considered CL problem, there exists an hypothesis space H' such that

$$\forall S \subseteq \mathcal{P}(X) \quad \exists h \in H' \mid S = \{x \in X \mid h(x) = 1\}$$

therefore H' can represent *any subset* of X. Now, consider the following:

- $\forall x' \notin X_D \quad \exists h', h'' \in VS_{H',D} : \begin{cases} h'(x) = 1 \\ h''(x) = 0 \end{cases}$ because H' can represent any subset $S \subseteq X$ — therefore, for *all* inputs outside X_D, a system using H' would not be able to perform a prediction;

- $\exists x' \notin X_D \mid \exists h', h'' \in VS_{H,D} : \begin{cases} h'(x) = 1 \\ h''(x) = 0 \end{cases}$ because H represents some subsets $S \subseteq X$ — therefore, for *some* inputs outside X_D, a system using H would not be able to perform a prediction;

- $h^* \in \arg\max_{h \in H} P(h, D)$ is such that for all $x' \notin X_D$, $h^*(x)$ is either 1 or 0 — therefore, for *all* inputs outside X_D a system using h^* would be able to perform a prediction.

Note that H' is the most powerful hypothesis space, because it can represent *any subset* of X, H is less powerful than X because it can represent *some subsets* of X, and h^* will only represent *one* subset of X, meaning that it is the least powerful representation. However, the more information the hypothesis space encapsulates about the values in X_D, the harder it becomes to **generalize** and **predict** values for samples *outside X_D*. In other words, a more expressive hypothesis space can **overfit** to the data, making it more difficult to make accurate predictions on unseen data.

Example 1.3 (Representation issues)**.** Consider the CL problem of enclosing all integer points on a 2D plane *labeled* with a $+$, thus

- X is the set of integer points on a 2D plane *labeled* with $+$ or $-$;

- Y is $\{+, -\}$;

- D is a set of pairs (p, y) where p is an integer point, and y is its label.

Consider the hypothesis space H that is composed of all the rectangles in the 2D plane with edges parallel to the axes; depending on the configuration of the points in X_D, H may not be able to enclose all the points with a $+$ in X_D. Now, consider the hypothesis

space H' such that each element in H' is the *union* of the region enclosed by multiple rectangles with edges parallel to the axes; this secon hypothesis space is clearly more powerful, because it can represent *any* possible configuration of the input points with a + in X_D.

However, for any given point x' that *is not* in X_D:

- it is *always* possible to find two elements in $VS_{H',D}$ wich will disagree whether x' has a + label or not;

- there *may* be two elements in $VS_{H,D}$ which will disagree whether x' has a + label or not;

- given h^*, it is *always* possible to *predict* whether x' has a + label or not.

In machine learning, the concept of **learning bias** is crucial for improving a model's ability to generalize. A good learning bias helps guide the learning algorithm towards patterns in the data that are useful for predicting unseen samples, increasing the system's generalization power. This bias allows the model to make accurate predictions on new data that wasn't part of the training set. Without such a bias, a system would simply **memorize** the dataset, failing to predict values for samples outside the training set, rendering it *ineffective* in real-world applications. Systems lacking generalization capabilities would be of little use, as they wouldn't be able to provide meaningful predictions beyond the data they were trained on.