richardwu.ca

CS 485/685 Course Notes

Machine Learning: Statistical and Computational Foundations

Shai Ben-David • Winter 2019 • University of Waterloo

Last Revision: January 10, 2019

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. These notes are my interpretation and transcription of the content covered in lectures. The instructor has not verified or confirmed the accuracy of these notes, and any discrepancies, misunderstandings, typos, etc. as these notes relate to course's content is not the responsibility of the instructor. If you spot any errors or would like to contribute, please contact me directly.

1 January 8, 2019

1.1 What is machine learning?

In machine learning, we aim to construct a program that takes as input **experiences** and produces as output **expertise**, or what we have learned from the experience.
We can then apply the **expertise** to produce useful programs such as a spam filter.
An example of learning in nature is **bait shyness**: rats who become sick from eating poisoned bait will become more cautious of food of similar characteristic in the future. Since rats will become more cautious of bait in the future, a delayed poison mechanism (rat is poisoned only 2 days after consuming the bait) is necessary for effective bait by de-associating poison from the bait.
Another example is an experiment called **pigeon superstition** by Skinner (1947): pigeons are starved in a cage with various objects. At random intervals, food is dispersed to satiate the pigeons. Eventually, each pigeon develops a "superstition": they each associate one arbitrary behaviour (e.g. a specific object or a specific movement) that results in food being dispersed.
On the contrary, Garcia (1996) tried a similar experiment to bait shyness with rats where poisoned and un-poisoned bait were identical in characteristic. Whenever a rat approached poisoned bait, a stimulus (e.g. bell ringing, electric shock) was applied to the rat. Surprisingly, the rats did not associate the arbitrary stimulus to the poisoning. This is contrary to the pigeon superstition: this can be explained by evolution (future generations are those that can become aware of poisonous bait) and the fact that rats have **prior knowledge** that poisoning comes from the bait itself, not some arbitrary stimulus.

1.2 Why do we need machine learning?

We desire machines to perform learning because machines can **process lots of data** and are (generally) **fast**.
We desire machines to *learn* because some tasks are simply to complex to hardcode in (e.g. image recognition). Some tasks we do not fully understand how to solve with hardcoded rules. Furthermore, learning allows adaptivity where the machine can constantly learn from new experiences and inputs.

1.3 Types of machine learning

Supervised and unsupervised Machine learning can be generally classified as either **supervised** or **unsupervised**.

Supervised learning takes labelled examples as experience and tries to re-produce these labels on future examples by learning rules. Spam detection may be supervised learning.

Unsupervised learning does not require labelled training data. Examples of unsupervised learning is outlier detection and clustering.

Semi-supervised learning takes as input both labelled and unlabelled data and sits between supervised and unsupervised.

Reinforcement learning also sits between supervised and unsupervised: the machine knows only the rules of the environment and takes actions until a reward (i.e. label) is produced. The machine then learns to label intermediary actions to the final reward produced in the episode (sequence of actions that resulted in the reward).

Passive and active We can also distinguish between **passive** and **active** learning: the former simply takes observed data whereas the latter involves actively performing experiments and interpreting the consequences of the experiments.

Teacher Machine learning can be guided by a "teacher" i.e. how the random sample used as input is generated. Teachers may be **indifferent**, **helpful** or **adversarial**. Helpful teachers produce hints and try to guide the program in the right direction whereas adversarial tries to fool the program.

Batch and online **Batch** learning is learning from a relatively large corpus of data before producing expertise. In contrast **online** learning requires the program to learn as experience is streamed and may result in more mistakes being made.

2 November 10, 2019

2.1 Components of a model

For example sakes, suppose we observe a number of papayas and assign them a score $\in [0, 1]$ for color and hardness. We then label each one as either tasty or not tasty. Using our observations, we would like to predict in the future the tastiness of papayas based on their color and hardness score.

Input The **input** to our learner consists of three parts:

Domain set (X) It is the set of our explanatory variates, in this case $[0, 1] \times [0, 1]$ corresponding to the color score and the hardness score.

Label set (Y) It is the set of our labels: tasty and not tasty.

Training set Our observations i.e. $\{(x_1, y_1), \ldots, (x_m, y_m)\} \subset X \times Y$

Output The **output** of our learner is a **prediction rule** $h : X \to Y$ i.e. the function we learn that maps our papaya scores to a label.

Simple generating model There exists some underlying (unknown) generating process of the population we are interested in (papayas). There is some unknown **probability distribution D over X** and some unknown **labelling rule** $f : X \to Y$.

Together (D, f) describes the generation of papayas.

Success measure We define some metric to measure how well our learner learns the underlying generating model. For example

$$L_{(D,f)}(h) = Pr_{x \sim D}\big[h(x) \neq f(x)\big]$$

We would like to minimize $L_{(D,f)}(h)$ to find the optimal learner h.

2.2 Empirical Risk Minimization (ERM)

As a first strategy, a learner can employ **Empirical Risk Minimization (ERM)** whereby it picks an h that minimize the errors on the *training sample*.

There is however an issue with this strategy: suppose we learn a rule where we label tasty for papayas with scores that exactly match our tasty papayas' scores and not tasty for everything else. That is

$$h_S(x) = \begin{cases} \text{tasty} & \text{if } (x, \text{tasty}) \in S \\ \text{not tasty} & \text{otherwise} \end{cases}$$

We define the **empirical loss (risk)** over a sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$

$$L_S(h) = \frac{|\{i \mid h(x_i) \neq y_i\}|}{m}$$

Note the above strategy give us exactly zero empirical error on our sample set S for any generating process but is obviously not a very robust strategy as it **overfits** to our sample.
Suppose the generating process is such that D is the uniform distribution over $[0,1] \times [0,1]$ and let

$$f(x_1, x_2) = \begin{cases} \text{tasty} & \text{if both coordinates in } [0, 1, 0.9] \\ \text{not tasty} & \text{otherwise} \end{cases}$$

Note that the empirical risk is $L_S(h) = 0$, but the risk on our population is $L_{(D,f)}(h) = (0.8)^2 = 0.64$ (since we would be predicting incorrectly for infinitely many points in the region $[0.1, 0.9] \times [0.1, 0.9]$).

2.3 Introducing prior knowledge with inductive bias

For the papaya example above, we could incorporate some prior knowledge such that tasty papayas belong in some rectangular region of color and hardness scores (which we must learn). We could have also assumed the tasty papayas belong in some linear halfspace, or some arbitrary region that we can learn.
More formally, prior knowledge are a set of rules H (set of functions from XS to Y) assumed by the learner to contain a good predictor.
Reformulating our previous ERM strategy, ERM_H picks $h \in H$ that minimizes empirical risk over the training set, that is we pick h^* where

$$h^* \in \text{argmin}_{h \in H} L_S(h)$$

Under some assumptions ERM_H has good success guarantees

Assumption 1: Realizability $\exists h \in H$ such that $L_{(D,f)}(h) = 0$

Assumption 2 S is picked iid by D and labelled by f (i.e. our sample is representative)

Theorem 2.1. Let H be a finite set of predictors. Assume our two assumptions hold. Then every ERM_H learning rule is guaranteed to converge to a zero-loss predictor as the sample size tends to infinity.
Namely for every ERM_H learner A and every $\epsilon > 0$

$$Pr_{S||S|=m}\big[L_{(D,f)}(A(S)) > \epsilon\big] \to 0$$

(this is exactly convergence in probability where $Pr\big[L_{(D,f)}(A(S))\big] \to 0$).

Proof. Let B_ϵ denote the set of all hypotheses in H that have error $> \epsilon$ i.e.

$$B_\epsilon = \{h \in H \mid L_{(D,f)}(h) > \epsilon\}$$

Let our set of "ugly" samples be

$$S_{ugly} = \{S \mid \exists h \in B_\epsilon \text{ s.t. } L_S(h) = 0\}$$

(samples where we have a zero empirical loss but has $> \epsilon$ loss on the true population).
Note that

$$Pr_S\big[L_{(D,f)}(A(S)) > \epsilon\big] \leq Pr_S\big[S_{ugly}\big]$$

that is the probability that our sample does not perform better than ϵ on our true population is bounded by the probability of picking an ugly sample (this is not just an equality since we also have samples S where $L_{(D,f)}(A(S)) > \epsilon$ and $L_S(A) > 0$). □