An Introduction to Machine Learning

Joseph Tenini

July 5, 2024

1 What is ML?

The answer to the question "What is ML?" will undoubtedly depend on who you ask. Instead of trying to provide a completely coherent language or ontology for ML, I will merely attempt to provide a descriptive one that gives you a mental framework for mapping tools to problems. If you find yourself observing that certain things that I have described as different are really special cases of each other - congratulations and your welcome. These are after all my favorite type of observations.

/subsectionThree tasks

I will describe machine learning as falling into three tasks:

1. Describing a labelled structure in data (supervised learning - SL).
2. Describing an unlabelled structure in data (unsupervised learning - UL).
3. Building agents that learn from experience (reinforcement learning - RL).

Before diving into more formal definitions - let's consider (lots of) examples. Imagine that you run a website dedicated to a two player board game you invented called snorg. Your website has it all - articles that are monitized by ads, the ability for users to play against a computer player, and a premium learning mode that gives players snorg problems that match their ability. Here are some problems you might encounter:

1.0.1 Looking for bots

While your site gets some great users, there are also a lot of bots that crawl your site doing various things. How can you identify these we sessions even if there is no explicit "bot label" in the data? (UL)

1.0.2 Looking for fraud

Some users sign up for your premium service only to pay with stolen credit cards that are later marked as fraudulent transactions by the bank. Can you flag these sign-ups as they happen so as not to be party to this bad behavior?

1.0.3 Recommending ads

Part of your site's revenue is from advertisements, where you are payed for each click. Which ad should you show to each user to maximize ad revenue? (RL)

1.0.4 Recommending articles

Since ads are shown alongside articles on your site, more articles read in each session translates into more revenue. At the bottom of each article you would like to recommend the three most similar articles. Given an article, what are the three most similar articles? (UL)

1.0.5 Predicting churn

What's the point of getting new premium service customers if you can't keep the ones you have? You would like to identify which customers are at risk of unsubscribing from the premium service so that you can offer them a discount to keep their business. Given a customer, what is the probability they will unsubscribe in the next 30 days without intervention? (SL)

1.0.6 Playing snorg

Since snorg is a bit of a rare game, it can be difficult to find a partner to play with. You need to create a snorg playing agent (with various difficulty levels) so that your users can play the game online at your site. (RL)

1.0.7 Representing a user

Each user at your site comes with a wealth of information - technographic data like device type and browser, temporal data like the date and time of their visits to the site, contextual data like what articles they read on the site or their snorg game details - can you represent users with a single vector of real numbers in an information dense way so that your models have a consistent and valuable feature set? (UL)

1.0.8 Creating an audience

Your advertisers will sometimes make ad buys based on audiences you create for them. They will upload a spreadsheet of known customers' email addresses. Your job is to identify these users within your site traffic and deliver users with similar interests or behaviors. (SL)

1.0.9 Selecting problems for a student

Your premium users have access to "snorg tutor" - a computor tutor that will choose snorg problems from a large pool. You need to design an agent that will select appropriate problems so that the user stays on the site as long as possible. (RL)

1.1 Supervised learning

Sometimes the universe grants us partial and noisy access to a function:

$$f : X \to Y$$

That is, for some subset $U \subset X$, we get noisy access to pairs $(u, n(f(u)))$ for some noise function n that fiddles with the truth (like randomness or measurement error). Our task is to learn f through some function approximation technique. I'll drop the business with n now to make the notation cleaner and be able to just refer to observing f directly, but know that it is there.

When $Y = \{0, 1\}$, the task is called classification. When $Y = \mathbb{R}$, the task is called regression.

The general approach for these tasks is to make two important choices:

1. A space of functions F that might contain something similar to f. 2. A notion of "distance" or "loss" L to f (really $n \circ f$) for F with respect to U.

That is, we need a space of functions F and a loss function $L_U(g, f)$ that tells us how much f and g disagree on U (or how much of the information of f is lost if we were to swap it out for g).

For a regression problem, this function L could be "squared error" - that is:

$$L_U(f, g) = \sum_{u \in U} (f(u) - g(u))^2$$

Given a choice of F, L, and U, we can search F to minimize L over U producing the "best fit" $\hat{f}$. A very useful approach to this strategy is starting at a randomly chosen function f_0 and moving successively in the direction of $-\nabla L_U(f, f_0)$ - a trick called gradient descent (if we compute L over only a small subset of U then it's called stochastic gradient descent).

Much of this course will be dedicated to understanding the choices of F and L as well as how to perform the search of the function space and how to understand the fitness of our candidate $\hat{f}$.

1.2 Unsupervised learning

I've been doing data science for about 10 years, and unsupervised learning is still a terrifying and confusing proposition. Essentially, it amounts to this: You recieve N samples from some random d-dimensional vector X - let's call them $x1, \ldots, x_N$. From this, you are left to describe $P(X)$. Since d is usually pretty big, the prospect is quite scary in general. The only joy to be found in the unsupervised learning life is by reducing the complexity of the problem some way. Let's consider two common ways that unsupervised learning naturally arises.

1.2.1 EDA for a supervised learning task

Suppose you are presented with a supervised learning task - predict y from x. One common first step is to learn about your predictive features x - that is, to

describe $P(X)$. As we have said, this is typically a frightening proposition, so we could proceed instead by simplifying space.

One natural simplification is to consider each coordinate of x independently - this is something we do naturally as humans. I recently met some of my coworkers in real life (after working with them for a year over zoom). I immediately projected these people down to a single number - height. Oh my goodness, this person is tall! These 4 are average, and this person is a bit short, and so on. We can do this with our feature space as well. Very often we will learn some interesting things that we can leverage or correct for in our further analysis. For example, if I noticed that one of my coworkers was 170 meters tall, I could check to see if a type had been made and make a correction in the data. After this, we can move on to analyzing how pairs of features move together - correlation analysis! Indeed, there are many simplifications of $P(X)$ that are worth considering when performing standard exploratory data analysis.

1.2.2 Finding "types"

Another common task a working data scientist encounters in the real world is to be presented with a data set and asked to catagorize data points by some "types" that need to be invented by the data scientist. For businesses, this could be categorizing customers into "audiences" or products into categories. Implicit in the task is the idea that the data is generated by a handful of independent distributions - one for each type. Perhaps your space of customers exists in three big clump, and one of the clumps has a very low lifetime value because these people can't understand the product! These are the sorts of things the person requesting the analysis has in mind when they ask for it, and it is undoubtedly an unsupervised learning task.

1.3 Reinforcement learning

Reinforcement learning generally refers to the practice of learning "good" decision making agents within the context of a Markov decision process. In this setting, an agent is presented with a state, must select an action, and then recieves as a result of the action, a new state and a real number reward. This continues either indefinitely or for a finite number of steps called an episode. Good agents are those that maximise the (possibly discounted) sum of rewards (called the return). The Markov property states that the mechanics that generate the new state and reward depend only on the current state and the action select (though is perhaps non-deterministic). That is, it is memory-less with respect to previous states and actions.

This is a very appealing set up because it is highly general and revolves around the concept of decision making. Indeed, performing analysis (like in unsupervised learning) and learning models of the world (like in supervised learning) are typically of interest only when they lead to better decision making. The RL perspective approaches this task wholistically. While general RL problems often prove very difficult, there is broad success in two cases:

1. The bandit problem where the MDP is simplified by having episode length of one (only one round of the state, action, reward dance)
2. Cases where access to the environment is very cheap and available at tremendous scale - game play like chess, go, and starcraft.