CS 337: Artificial Intelligence

Amit Rajaraman

Last updated August 21, 2021

Contents

§1. Introduction

1.1. What and Why?

Machine learning involves learning from past experiences to perform a job better.
Broadly, the goal of the field is to make a computer emulate human pattern recognition, which is second nature to us. What do each of these words mean in the context of computer science? "Learning" and "better" usually depend heavily on the context surrounding our goals, and past experiences usually refers to a data set of some form. Broadly, there are two types of learning: *supervised* and *unsupervised*.

In the former, we have access to some **training data**, usually consisting of a set of pairs of input and output. The supervised learning algorithm then analyzes this data to produce an inferred function, which can then be used for determining what unknown inputs (not in the training data) must map to. This requires the algorithm to go from a smaller set of training data to a much broader set of inputs in a way that "makes sense". For example, given the heights and weights of a thousand people, we might be asked to determine the weight of a person of a person of some new height.
In unsupervised learning on the other hand, we do not have access to any prior information, so we must classify them in some sensible manner. For example, given a set of fruits, we may wish to classify them into various groups. If we divide on the basis of colour (alone), we might get two groups – apples and cherries in one and oranges and peaches in the other. If we divide on the basis of both colour and size, we might further divide the first of these groups into two. The computer is forced to create some compact internal representation of the features of these fruits, and from this we might even be able to generate *new* fruits.

How does all this work? It might be simpler to grasp if we dive headlong into an example.
Suppose we are given a data set $\mathcal{D} = \{\langle x_1, y_1\rangle, \ldots, \langle x_k, y_k\rangle\}$. We wish to determine a function f^* such that $f^*(x)$ is the best "predictor" of y with respect to $\mathcal{D}$.
To quantify how good a prediction is, we introduce an *error function* $\varepsilon(f, \mathcal{D})$ that reflects the discrepancy of the function with respect to $\mathcal{D}$. We also assume that our f^* is taken from some base class of functions, say $\mathcal{F}$. We then set

$$f^* = \arg\min_{f \in \mathcal{F}} \varepsilon(f, \mathcal{D}).$$

Typically, we present our function in terms of some "basis functions" $(\phi_i)_{i=1}^n$ each from the set of inputs to $\mathbb{R}$. More compactly, we have a single function ϕ from the set of inputs to $\mathbb{R}^n$ that shows all the relevant things we can glean about any input. Suppose our set of outputs is in $\mathbb{R}^k$. We then learn a function $f : \mathbb{R}^n \to \mathbb{R}^k$ and given a new input x', we predict the corresponding output as $y' = f(\phi(x'))$.
Due to this, it is often helpful to think of the input as an element of $\mathbb{R}^n$, where n is the number of basis functions.

Remark. It is important to note that even though our original input x may be in $\mathbb{R}$, we could have more than just 1 basis function. For instance, we could choose the basis functions as $\{(x \mapsto x^i) : 1 \leq i \leq 100\}$. This is especially important in the coming topic of linear regression, where linearity in the basis functions should not be mixed up with linearity in the original input.

1.2. Linear Regression

We first discuss perhaps the simplest example of machine learning. In this, the class $\mathcal{F}$ of functions we consider is merely the class of all linear functions over the basis functions. That is,

$$\mathcal{F} = \{(x \mapsto w^\top \phi(x)) : w \in \mathbb{R}^n\}.$$

Observe that if one of our basis functions is 1 (or some constant), this is equivalent to the set of all *affine* functions (linear functions plus a constant).

1.2.1. The method of least squares

Suppose our data set is $\mathcal{D} = \{\langle x_i, y_i\rangle : 1 \le i \le m\}$, where each y_i is in $\mathbb{R}$. In the method of least squares, our choice of error function is given by

$$\varepsilon(f, \mathcal{D}) = \sum_{i=1}^{m} (f(x_i) - y_i)^2.$$

The squared difference is an extremely common error function for various reasons – it is convex, non-negative, and well-behaved.
In the case of linear regression, where any function f is uniquely determined by a $w \in \mathbb{R}^n$, we can express the error in terms of this vector as

$$\varepsilon(w) = \sum_{i=1}^{m} (w^\top \phi(x_i) - y_i)^2.$$

Now, set

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_n(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_m) & \cdots & \phi_n(x_m) \end{pmatrix} \text{ and } y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}.$$

So,

$$\varepsilon(w) = \left\| \Phi^\top w - y \right\|_2^2.$$

The minimum value of the above is just the distance from y to $\mathcal{C}(\Phi)$, the column space of Φ! Let $w^* = \arg\min_w \varepsilon(w)$. In particular, if $y \in \mathcal{C}(\Phi)$, it is possible to get the cost to 0.
The least square solution is then the distance from y to the projection $\hat{y}$ of y on $\mathcal{C}(\Phi)$. How do we find $\hat{y}$ and w^*? The line joining y and $\hat{y}$ is orthogonal to $\mathcal{C}(\Phi)$. As a result, $(\hat{y} - y)^\top \Phi = 0$. Therefore,

$$\begin{aligned} \hat{y}^\top \Phi &= y^\top \Phi \\ (\Phi w^*)^\top \Phi &= y^\top \Phi \\ w^* &= (\Phi^\top \Phi)^{-1} \Phi^\top y. \end{aligned}$$

This is well-defined only if $(\Phi^\top \Phi)$ is invertible and Φ has full column rank (Why?).

Remark (Overfitting). One might think that more basis functions means a better approximation. Indeed, this would mean that we have more "freedom", which should allow us to get a better function. However, this is not the case. While adding more basis functions allows us to emulate the *training* data better, we might mimic it too closely and lose sight of the overall behaviour, which results in bad performance when it comes to the general *testing* data.
This is very clearly seen in the following example where the output is slightly noisy linear data, and we have taken two cases: one wherein the basis function are just $\phi(x) = (1, x)$ (fitting a degree 1 polynomial), and the second where $\phi(x) = (1, x, \ldots, x^{10})$ (fitting a degree 10 polynomial).

FIGURE 1 – Overfitted data