

# Chapter - 1

## Learning from Data

Aviral Janveja

### 1 Introduction

Machine Learning is a field concerned with learning patterns from data without explicitly programming them. For example, we learn to recognize trees not by memorizing a formal definition, but by repeatedly seeing them in real life or in pictures. Over time, we learn to identify their distinct physical features like shape or color. Similarly, machine learning systems do not require explicit programming, they learn by identifying patterns directly from data.

Before going any further, it is essential to ask, when does it make sense to use machine learning and when does it not? For machine learning to serve as a practical solution to a problem, the following conditions must be met :

#### 1.1 Existence of a Pattern

Machine learning is fundamentally about uncovering patterns from data. If a certain process is completely random or unpredictable, no algorithm will be able to extract any meaningful structure out of it.

#### 1.2 Mathematical Infeasibility

When a problem can be expressed precisely in mathematical or programmable terms, for example, dividing two numbers, then there is no need for learning from data.

However, for many real-world tasks, such as predicting a user's movie preferences, it is impractical to define an exact formula. In such cases, learning from data is the natural approach.

#### 1.3 Availability of Data

Since machine learning is about learning from data, having access to sufficient and relevant data points is a non-negotiable requirement. Without data, the learning process cannot even begin.

## 2 The Machine Learning Process

Now, let us formalize the machine learning process through a simple example from the financial domain. Suppose a customer applies for a loan. The bank must decide

whether granting the loan is a good idea or not, since its goal is to maximize profit while minimizing risk.

Naturally, there is no magic formula that can perfectly predict whether a customer will be creditworthy. Instead, the bank relies on historical records of past customers and how their repayment behavior turned out in hindsight. The idea is to identify patterns through these past data-points to help predict the creditworthiness of future applicants. With this example in mind, we can now outline the formal components of the machine learning process.

## 2.1 Target Function

The target function represents the underlying pattern that we are looking to uncover. It is essentially unknown and the learning model tries to approximate it through the available data-points. Formally, it is denoted as :

$$f : X \rightarrow Y$$

Here  $X$  is the domain of the function, that is the set of all possible inputs and  $Y$  the codomain, is the set of all possible outputs.

## 2.2 Data

The data from say,  $N$  previous customers is therefore represented as :

$$\text{Data} : (x_1, y_1), (x_2, y_2), (x_3, y_3) \dots (x_N, y_N)$$

The **Input**, which is the customer application information such as age, salary, years in residence, current debt and so on. Represented by a column matrix  $x$ . The **Output**, which captures whether the customer was credit worthy or not in hindsight, from the bank's point of view. Represented by a binary  $y$ .

## 2.3 The Learning Model

The learning model specifies the form of the hypothesis function  $h$ . There are  $m$  candidate hypotheses in total forming the hypothesis set :

$$H = \{h_1, h_2, h_3 \dots h_m\}$$

Out of which a final hypothesis function is chosen that is expected to best approximate the target function. This final hypothesis function is chosen from the hypothesis set based on the available data-points. The data points are fed into the learning model, which then searches through a hypothesis set  $H$  and comes out with the final hypothesis function :

$$h_f : X \rightarrow Y$$

The final hypothesis function is our approximation of the unknown target function  $f$ . The goal in machine learning is that  $g$  approximates  $f$  well, that is  $g \approx f$ .

In other words, the learning model acts as the bridge between the data and the final hypothesis. In the next section, we analyze one of the simplest learning models: the Perceptron.

learned through the available data points, it is selected from a set of candidate hypotheses provided by the learning model.. It is denoted by :

### 3 Perceptron

Continuing with the loan example from above, let us begin by looking at the given data, which is available to us in the form of  $N$  input-output pairs :

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3) \dots (\mathbf{x}_N, y_N)$$

Here  $y_n$  is equal to +1 or -1 representing whether the customer was credit-worthy or not in hindsight and  $\mathbf{x}_n$  is the application information of the  $n^{th}$  customer, which is represented by a column matrix as shown below :

$$\mathbf{x}_n = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

Here  $x_1, x_2$  and so on, upto  $x_d$  are the attributes of a customer like age, salary, years in residence, outstanding debt and so on.

#### 3.1 Perceptron Function

Now, what a perceptron basically does, is to assign weights to each of the customer attributes. These weights are multiplied to their respective attributes and then summed-up as follows :

$$w_1x_1 + w_2x_2 + \dots + w_dx_d$$

The above linear summation of weights and attributes is called the **credit score** of an individual customer. The idea is pretty simple, If the credit score is greater than a certain threshold, then we approve the loan, else we deny it :

$$w_1x_1 + w_2x_2 + \dots + w_dx_d > \text{threshold (Approve Loan)}$$

$$w_1x_1 + w_2x_2 + \dots + w_dx_d < \text{threshold (Deny Loan)}$$

This is equivalent to :

$$w_1x_1 + w_2x_2 + \dots + w_dx_d - \text{threshold} > 0 \text{ (Approve Loan)}$$

$$w_1x_1 + w_2x_2 + \dots + w_dx_d - \text{threshold} < 0 \text{ (Deny Loan)}$$

We can simplify the above expression further, by taking the minus of threshold as the zero<sup>th</sup> weight ( $w_0 = -\text{threshold}$ ) and introducing an artificial attribute  $x_0 = 1$  to go along with it, hence giving us :

$$w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d > 0 \text{ (Approve Loan)}$$

$$w_0x_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d < 0 \text{ (Deny Loan)}$$

Next, we re-write the above expression in vector (column matrix) notation to make it shorter and cleaner :

$$w_0x_0 + w_1x_1 + \dots + w_dx_d = \begin{bmatrix} w_0 & w_1 & \dots & w_d \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} = \mathbf{w}^T \mathbf{x}$$

The above formula outputs a real value that could be greater than, equal to or less than zero. In order to output a binary value, similar to  $y$ , we can wrap the above formula inside a **sign** function.

$$\text{sign}(\mathbf{w}^T \mathbf{x})$$

The sign is just a simple function that takes in a real number value and outputs +1 if the value is greater than zero and -1 if it is less than or equal to zero, hence giving us our final credit decision.

### 3.2 Perceptron Learning Algorithm

Looking at the perceptron function, that we have arrived at :

$$g(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x})$$

We see that  $\mathbf{x}$  representing the customer information is already a given, along with  $y$ , which is the correct credit decision in hindsight. So, the underlying pattern that we are trying to learn here, will be reflected in our choice of weights, as represented by the weight vector  $\mathbf{w}$ . Hence, the task is to utilize the given data points to arrive at a set of weights, that are hopefully able to classify future customers well, that is :

$$g(\mathbf{x}) \approx f(\mathbf{x})$$

So, how do we arrive at a suitable set of weights? First, we initialize the weights randomly, let us say, set them all to 1. The perceptron learning algorithm then iterates through the dataset, looking for misclassified points  $(\mathbf{x}_n, y_n)$ , where :

$$g(\mathbf{x}) \neq f(\mathbf{x})$$

that is,  $\text{sign}(\mathbf{w}^T \mathbf{x}_n) \neq y_n$

Whenever such a misclassified point is found, the weight vector is updated to nudge the classification decision in the right direction by applying the following **update rule** :

$$\mathbf{w}_{new} = \mathbf{w} + y_n \mathbf{x}_n$$

This process continues, until no misclassified points remain. And that's it, we then have our required set of final weight values.

### 3.3 Justification for the Update Rule

If a point  $(\mathbf{x}_n, y_n)$  is misclassified, it means :

$$\text{sign}(\mathbf{w}^T \mathbf{x}_n) \neq y_n$$

Accordingly, the perceptron updates the weight vector as follows :

$$\mathbf{w}_{new} = \mathbf{w} + y_n \mathbf{x}_n$$

Let us examine how this update affects the classification decision. Substituting the updated weight vector from above, we get :

$$\mathbf{w}_{new}^T \mathbf{x}_n = (\mathbf{w} + y_n \mathbf{x}_n)^T \mathbf{x}_n$$

Expanding the right-hand side, we get :

$$\mathbf{w}_{new}^T \mathbf{x}_n = \mathbf{w}^T \mathbf{x}_n + y_n \mathbf{x}_n^T \mathbf{x}_n$$

This is equivalent to :

$$\mathbf{w}_{new}^T \mathbf{x}_n = \mathbf{w}^T \mathbf{x}_n + y_n \|\mathbf{x}_n\|^2$$

**Therefore**, for a misclassified point, if  $\text{sign}(\mathbf{w}^T \mathbf{x}_n) = -1$ , then :

$$\mathbf{w}_{new}^T \mathbf{x}_n = \mathbf{w}^T \mathbf{x}_n + \|\mathbf{x}_n\|^2$$

And when  $\text{sign}(\mathbf{w}^T \mathbf{x}_n) = +1$ , then:

$$\mathbf{w}_{new}^T \mathbf{x}_n = \mathbf{w}^T \mathbf{x}_n - \|\mathbf{x}_n\|^2$$

Hence, showing that each update to the weight vector nudges the perceptron in the required direction, improving its classification decision on that particular data point. If the dataset was **linearly separable** to begin with, then perceptron learning algorithm is guaranteed to find the set of weights, that classify all the data-points correctly.

Data being linearly separable simply means that, when the data points are plotted on a two-dimensional graph, it is possible to draw a straight-line that separates them into two distinct categories. The following image illustrates the above discussion :

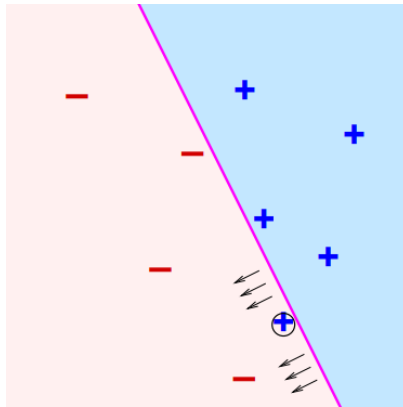


Figure 1: Perceptron

The purple line represents our linear perceptron classifier that is uniquely determined by the choice of weight vector  $w$ . We have a misclassified **plus** point in the **minus** region and the update-rule tries to correct this by nudging the classifier in the required direction.

## 4 Types of Learning

As discussed in the introduction, machine learning is about learning patterns from data. Alternatively, we can put it as **using a set of observations to uncover an underlying process**. In either case, we have established that having the set of observations or data-points is a non-negotiable. However, the form in which the data is available to us can vary. Hence, based on the form of available data points and the type of learning model employed, machine learning can be broadly categorized into three types : supervised learning, unsupervised learning and reinforcement learning.

### 4.1 Supervised Learning

Supervised learning is when a model is provided with correctly labeled data (**input, correct output**) as seen in the perceptron example above ( $x_n, y_n$ ).

### 4.2 Unsupervised Learning

Unsupervised learning is when the model is required to find patterns or clusters within an unlabeled dataset (**input, no output**).

### 4.3 Reinforcement Learning

Reinforcement learning is when the model learns from the consequences of its actions (**input, score for a particular choice of output**).

## 5 References

1. CalTech Machine Learning Course - CS156, Lecture 1.