Chapter - 1
Learning from Data

Aviral Janveja

1 Introduction

Artificial Intelligence refers to the ability of machines, to perform tasks that typically require human intelligence, such as learning, reasoning, visual recognition, use of language and so on. Machine Learning on the other hand is a major subset of AI, that deals with the development of algorithms that can learn patterns from data. For example, we learn to recognize trees just by looking at them repeatedly, either in real life or through pictures. Essentially, we are learning to recognize their distinct physical features through visual data, since childhood. We don't really require some exact definition of a tree in order to recognize one, just a picture is enough. This is basically the idea behind machine learning or "learning from data".

Before diving into the formal process of machine learning, 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 three important conditions must be met :

1.1 A pattern exists

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

1.2 It is unfeasible to pin it down mathematically

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

However, in 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 We have the 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 an example from the financial domain. Consider a customer who applies for a loan. The bank needs to decide whether it is a good idea to extend the loan or not, as the bank wants to maximize its profits.
Naturally, the bank does not have a magic formula to forecast, if a customer will be credit worthy or not. Instead, they are going to rely on the historical records of previous customers and how their behavior turned out to be in hindsight. The **idea** is to look for a pattern in those past data points, in order to predict the credit worthiness of future customers. Let us look at the formal components of the machine learning process next :

2.1 Target Function

The target function represents the underlying pattern that is essentially unknown. We try to learn it approximately, through the available data points. It is denoted by :

$$f : X \to Y$$

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 historical data of past customers, consists of two parts :

1. 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 $\mathbf{x}$.
2. 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.

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

$$\textbf{Data} : (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), (\mathbf{x}_3, y_3)...(\mathbf{x}_N, y_N)$$

As discussed before, these data points are used to uncover the underlying pattern relating customer data to their credit worthiness.

2.3 Hypothesis Function

The hypothesis function is the formal name for our approximation of the unknown target function f, learned through the available data points. It is denoted by :

$$g : X \to Y$$

The goal in machine learning is that g approximates f well, that is $g \approx f$.

2.4 The Learning Model

Finally, the learning model is what connects the data to the hypothesis function. The data points are fed into the learning model, which in turn comes out with the final hypothesis function. We analyze a simple learning model, called Perceptron next.

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 :

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

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 :

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

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 :

$$w_1x_1 + w_2x_2 + ... + w_dx_d$$

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 :

$$w_1x_1 + w_2x_2 + ... + w_dx_d > \text{threshold (Approve Loan)}$$
$$w_1x_1 + w_2x_2 + ... + w_dx_d < \text{threshold (Deny Loan)}$$

This is equivalent to :

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

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

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

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

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

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.

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

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 :

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

We see that 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 **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 :

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

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 :

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

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 :

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

Expanding the right-hand side, we get :

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

This is equivalent to :

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

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

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

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

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

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 :

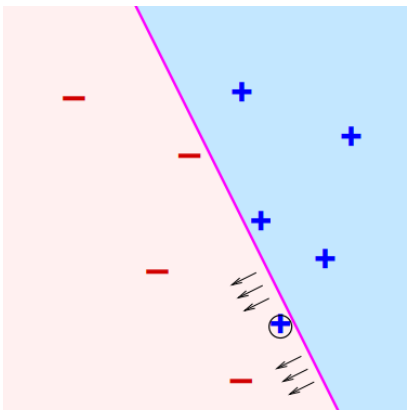

Figure 1: Perceptron

The purple line represents our linear perceptron classifier that is uniquely determined by the choice of weight vector $\mathbf{w}$. 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 ($\mathbf{x}_n$, 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.