Chapter - 1
Learning from Data

Aviral Janveja

1 Introduction

Artificial Intelligence refers to the ability to perform tasks that typically require human intelligence, such as learning, reasoning, problem-solving and decision-making. Machine Learning is a major subset of AI that is focused on the development of algorithms that can **learn the underlying patterns from data without requiring explicit programming**. The title of the first chapter : learning from data, thus faithfully describes what the subject is about.

2 Pre-requisites for Machine Learning

If you find these three components in a problem statement, then machine learning is ready as an application tool.

2.1 A pattern exists

This condition makes sense because if there is no pattern, just unpredictable randomness, then there is nothing to learn.

2.2 It is unfeasible to pin it down mathematically

If the nature of problem is such that you can program it precisely and easily in mathematical terms, for example dividing two numbers, then you do not require learning from data.
However, for instance, when trying to determine a user's movie preferences, it is easy to see that writing an exact mathematical formula for that is unfeasible. Learning from data, makes more sense in such scenarios.

2.3 We have the data

As discussed, we are learning from data. So we first and foremost need to have the relevant data points. Otherwise, machine learning is not applicable.

3 The Machine Learning Process

Next, let us formalize the machine learning process through an example from the finance 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 decide whether a customer is credit worthy or not. Instead, they are going to rely on the historical records of customers and how their credit behavior turned out to be in hindsight. The idea is to look for a pattern in those data points, in order to predict the credit worthiness of future customers. The following are the major components of the machine learning process :

3.1 Data

The data, which in this case refers to the historical record of 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 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 historical 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.

3.2 Target Function

This underlying pattern obviously, is not exactly known. Hence, we try to learn it approximately through the available data points. It is formally called the target function and denoted by :

$$f : X \to Y$$

Here X is the domain of f, that is the set of all inputs from the data above and Y the codomain, is the set of all outputs.

3.3 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 $g \approx f$.

3.4 The Learning Model

The learning model is what connects the data-set to the hypothesis function. The data points are fed into the learning model, which in turn comes out with the final hypothesis function. As an example, we analyze a simple learning model, called Perceptron next.

4 Perceptron

Continuing with our example from the finance domain, let us begin by looking at the given data. The data 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 $y_n \in \{+1, -1\}$ represents the binary decision of approving or rejecting the loan request and $\mathbf{x}_n$ is the application information of the n^{th} customer, which is represented by a column matrix as shown below :

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

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

4.1 Perceptron Function

What perceptron does, is give different weights to each of the customer attributes. We multiply weights with their respective attributes and then add them together in a linear form :

$$w_1x_1 + w_2x_2 + ... + w_dx_d = \sum_{i=1}^{d} w_ix_i$$

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. This can be written as :

$$\sum_{i=1}^{d} w_ix_i > \text{threshold (Approve Loan)}$$
$$\sum_{i=1}^{d} w_ix_i < \text{threshold (Deny Loan)}$$

This is equivalent to :

$$\sum_{i=1}^{d} w_i x_i - \text{threshold} > 0 \text{ (Approve Loan)}$$

$$\sum_{i=1}^{d} w_i x_i - \text{threshold} < 0 \text{ (Deny Loan)}$$

The above formula outputs a real value. In order to output a binary value (+1 or -1) representing approval or denial of credit respectively, similar to the output y of our data points, we can wrap the above formula inside a **sign** function.

$$\text{sign}\left(\sum_{i=1}^{d} w_i x_i - \text{threshold}\right)$$

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

Further, we can simplify the above notation by making the threshold as just another weight w_0 and introduce an artificial coordinate $x_0 = 1$ to go along with it. This simplifies the formula to :

$$\text{sign}\left(\sum_{i=0}^{d} w_i x_i\right)$$

The above can be re-written in the vector (column matrix) notation as follows :

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

Here, $\mathbf{x}$ is the column matrix representing the various attributes of a customer and $\mathbf{w}^T$ is the transposed matrix, representing the weights attached to each attribute. Hence giving us our final perceptron function :

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

4.2 Perceptron Learning Algorithm

Given the set of data points from existing customers :

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), (\mathbf{x_3}, y_3)...(\mathbf{x}_N, y_N)$$

The perceptron learning algorithm iterates through the dataset, looking for a misclassified point $(\mathbf{x}_n, y_n)$, that is :

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

When a misclassified point is found, the weight vector is updated to nudge the classifier 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 in a full pass through the dataset.

4.3 Justification for the Update Rule

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

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

That is, y_n was +1 but sign$(\mathbf{w}^T \mathbf{x}_n)$ gave the output -1 or vice versa. This implies that :

$$y_n \ . \ \text{sign}(\mathbf{w}^T \mathbf{x}_n) \leq 0$$

Now, the perceptron updates the weight vector as follows :

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

Therefore, let us see how this update affects the linear product in perceptron. Substituting the values 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 :

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

This is equivalent to :

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

Multiplying with y_n on both sides, we get :

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

Ofcourse, $y_n^2 = 1$ as $y \in \{+1, -1\}$, hence :

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

Hence, the update ensures that :

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

This shows that, each update to the weight vector nudges the perceptron in a direction that improves its classification on that particular data point. Further, as per the perceptron **convergence theorem**, if the dataset was **linearly separable** to begin with, then the perceptron learning algorithm is guaranteed to find the weight vector that correctly classifies all of the data points.

The proof for the convergence theorem is not provided here. And the data being linearly separable simply means that, when the data points are plotted on a two dimensional plane, it is possible to draw a line that correctly separates them in two categories. The following image illustrates the above discussion :

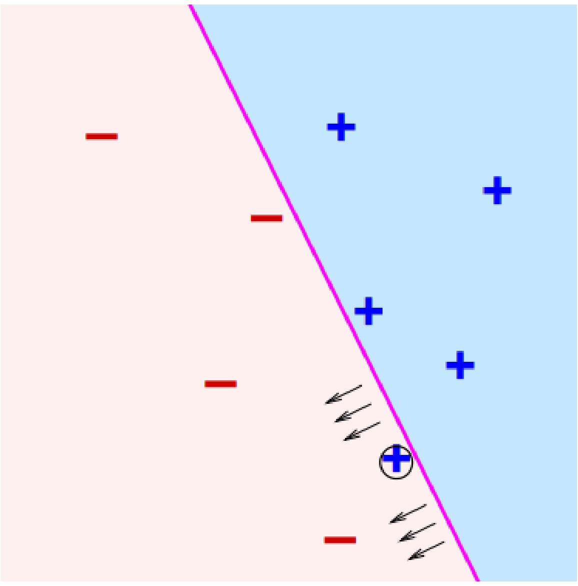

Figure 1: Perceptron

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 it by nudging the classifier in the right direction. Each update gradually rotates the decision boundary to improve classification. If the data is linearly separable, the perceptron will eventually stop updating once no misclassified points remain.

5 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 is a non-negotiable. However, 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.

5.1 Supervised Learning

Supervised learning is when a model is provided with labeled data (**input, correct output**) as seen in the perceptron example above ($\mathbf{x}_n$, y_n).

5.2 Unsupervised Learning

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

5.3 Reinforcement Learning

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

6 References

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