Chapter - 1
Learning from Data

Aviral Janveja

1 Introduction

Machine Learning is a field concerned with learning patterns from data, without having to explicitly program those patterns ourselves. For example, we learn to recognize trees not by memorizing their definition, but by repeatedly seeing them in real life or in pictures. Over time, we learn to identify their distinct physical features. Similarly, machine learning systems work by learning patterns directly from example data-points.

2 The Machine Learning Process

Next, let us formalize the process of learning from data, 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 or not. Instead, the bank relies on historical records of past customers and how their repayment behavior turned out to be in hindsight. The idea is to learn patterns from 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 Unknown Target Function

The target function represents the underlying pattern that we are looking to uncover. In our example, it is the unknown magic function that relates the applicant information to their creditworthiness. As this function is not known in exact mathematical or programmable terms, we try to learn it approximately through the available data-points. Formally, 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 is the codomain, the set of all possible outputs.

2.2 Data

The data from N past customers is given as :

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

Here $\mathbf{x}_n$ is a column matrix representing customer information such as age, salary, years in residence, current debt and so on. Whereas $y_n \in \{+1, -1\}$ is a binary value representing whether the customer was creditworthy or not in hindsight.

2.3 The Learning Model

The learning model defines the **hypothesis set** and the **learning algorithm**. The hypothesis set H is a set of hypothesis functions $\{h_1, h_2, h_3...h_m\}$ out of which the learning algorithm picks the **final hypothesis function** h_f based on the available data-points.

This final hypothesis function is our approximation of the unknown target function. The goal in machine learning is that h_f approximates f well, that is :

$$h_f \approx f$$

An an example, we examine a simple learning model called Perceptron next, to understand how the above process works, continuing with the bank-loan example.

3 Perceptron

Let us begin by looking at the given data, which is available to us in the form of N input-output pairs :

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

As discussed, $\mathbf{x}_n$ 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. Whereas $y_n \in \{+1, -1\}$ is binary value representing the creditworthiness of a past customer in hindsight.

3.1 Perceptron Hypothesis 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 can be 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 :

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

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 wrap the above formula inside a **sign** function.

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

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 the final form of our perceptron hypothesis function.

3.2 Perceptron Learning Algorithm

Examining the perceptron hypothesis function, that we have arrived at above :

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

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

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

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

that is;

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

Whenever such a misclassified point is found, the weight vector is updated 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 the dataset. And that is it, we then have our required set of weight values.

3.3 Update Rule Justification

Let us examine how the above defined update rule actually nudges the perceptron classifier in the right direction at each misclassified point. 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 :

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

Substituting the updated weight vector from above and ignoring the sign function for now, we get the following :

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

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, where $y_n = +1$ and $\text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_n) = -1$, it implies $\mathbf{w}^{\mathbf{T}}\mathbf{x}_n \leq 0$ and after the update we get :

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

The update is adding some positive value to $\mathbf{w}^{\mathbf{T}}\mathbf{x}_n$ which might make $\mathbf{w}^{\mathbf{T}}\mathbf{x}_n > 0$. Although, it does not guarantee an immediate sign flip, it clearly nudges the perceptron classifier in the right direction.

Similarly, when $y_n = -1$ and $\text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_n) = +1$, we get:

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

Hence, showing that each update to the weight vector nudges the perceptron in the required direction, improving its performance on that particular data-point. The following image illustrates the above discussion :

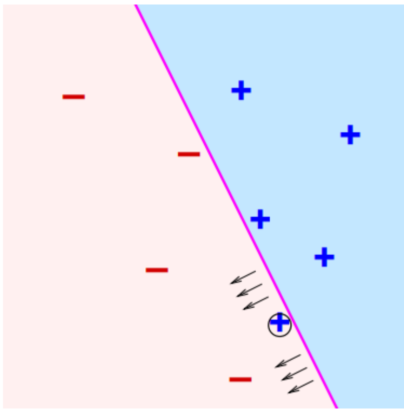

Figure 1: Perceptron

The pink 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.

3.4 Convergence Theorem

The convergence theorem for the perceptron guarantees that if the data is **linearly separable**, repeated updates will eventually classify all 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 data-points in the above image for example, are linearly separable.

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 a set of observations or data-points is a non-negotiable. However, the form in which the data is available to us can vary. Depending on this form and the nature of the learning task, machine learning is commonly divided into three main paradigms: supervised learning, unsupervised learning, and reinforcement learning.

4.1 Supervised Learning

In supervised learning, the model is trained on labeled data, where each example consists of an input and its corresponding correct output. The goal is to learn a function from inputs to outputs that generalizes well to unseen data. For instance, the perceptron example above used labeled pairs $(\mathbf{x}_n, y_n)$.

4.2 Unsupervised Learning

In unsupervised learning, the data consists only of inputs without associated labels (**input, no output**). The task is to discover hidden structures, patterns, or clusters within the data. Examples include clustering, dimensionality reduction, and generative modeling.

4.3 Reinforcement Learning

In reinforcement learning, the model interacts with an environment by taking actions and receiving feedback in the form of rewards or penalties (**action, feedback**). The model learns by trial and error to maximize its cumulative reward, gradually improving its decision-making. This setup is especially suited to sequential problems such as game playing, robotics and control systems.

5 References

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