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. Instead, the bank relies on historical records of past customers and how their repayment behavior turned out 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 Target Function

The target function represents the underlying pattern that we are looking to uncover. In our example, it would be the function that relates the applicant information to their creditworthiness. Obviously, such a function is never known in precise mathematical or programmable terms, which is why we try to learn it approximately through the available data-points. Formally, it is denoted as :

$$f : X \to Y$$

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 previous customers is represented as :

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

$\mathbf{x}_n$ is a column matrix representing the applicant information of the customer like age, salary, years in residence, current debt and so on.

$y_n = \{+1, -1\}$ is a binary value equal that represents whether the customer was creditworthy or not in hindsight.

2.3 The Learning Model

The learning model defines the representation of the hypothesis function h. All the candidate hypothesis functions together form the **hypothesis set** :

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

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 :

$$h_f : X \to Y$$

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 :

$$h_f \approx f$$

3 Perceptron

In this section, we analyze one of the simplest learning models, the Perceptron. Continuing with the bank-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 :

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

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 $y_n = \{+1, -1\}$ represents the creditworthiness of the 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 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 ($w_0 = -$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 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 the final form of our perceptron hypothesis function.

3.2 Perceptron Learning Algorithm

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

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

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 will be reflected in our choice of weights, as represented by the weight vector = $\mathbf{w}$. Hence, the task now is to utilize the given data-points to arrive at a set of weights, that are hopefully able to classify future loan-applicants well.

So, how do we compute the right set of weights? First, we initialize the weights randomly. The perceptron learning algorithm then iterates through the dataset, looking for misclassified points $(\mathbf{x}_n, y_n)$, where :

$$\begin{aligned} &h(\mathbf{x}) \neq f(\mathbf{x}) \\ &\text{or} \\ &\text{sign}(\mathbf{w}^{\mathbf{T}}\mathbf{x}_n) \neq y_n \end{aligned}$$

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 the required set of final weight values.

3.3 Justification for the Update Rule

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 :

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

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 :

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

(make this better and more detailed, add from sign to value of wtx less than or more than zero and then how perceptron adds or subtracts from it accordingly. say sign wtx is -1, whereas y is +1, which means wtx should have been more than zero, where it is less than zero, hence the weights are wrong and need to be updated. So after the update w new adds some positive value to wtx, hence trying to make it greater than zero and thus nudging it in the right direction, trying to make it agree with y = +1.)

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

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

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

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

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.

(add image here instead)

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 :

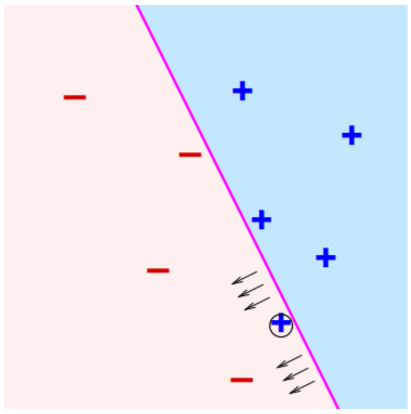

Figure 1: Perceptron

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 ($\mathbf{x}_n$, 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**). (can add some more info, like generative models)

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**). (can add some more clarification like trial and error with GPT help)

5 References

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