A Machine Learning Study

José António Portela Areia

January 2023

Part I
Introduction

The level of research in areas of study like machine learning and/or neural networks have been dramatic growing in the last couple of decades. The potential that these technologies have are great, however they are a little complex to understand and master. But it's now clear that machine learning offers a powerful set of tools for solving problems in areas like speech recognition, customer service, computer vision, patter recognition and many more.

Much of this brief report have the goal to expose a bit more about machine learning, neural networks, multilayer perceptrons and some usage metrics, like for example, confusion matrix, recall, accuracy and so on. In a final chapter it will expose and discussed some pre-processing techniques (e.g., encoding, normalization, etc.).

Part II
Machine learning

While a number of different definitions for machine learning (ML) have surface over the last couple of decades, TM Mitchell offers the following explanation in his 1997 book "ML studies algorithms that improve with (learn from) experience.". In order to explain the concept of 'learning a problem' a bit more, I will cite another quote from his book, where he explains the general definition of that concept "A computer program is said to *learn* from experience E with respect to some class of *tasks* T and *performance measure* P, if its performance a task in T, a measured by P, improves with experience E.".

With that said, the concept of machine learning is to learn from experience to achieve some prediction with that learning. The following *Figure 1* shows a general schema for machine learning methods.

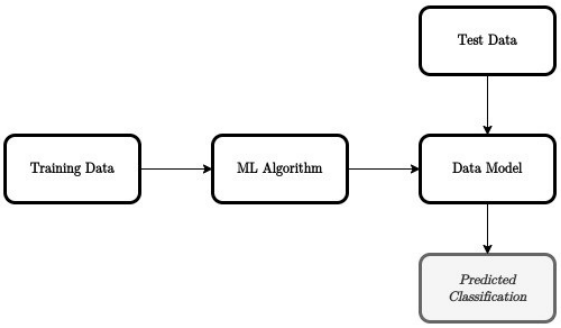

Figure 1: Schema for machine learning methods

1 Machine learning methods

A machine learning process can be divided in three distinct methods: supervised machine learning, unsupervised machine learning and semi-supervised machine learning. Below lies a brief explanation of the methods presented above.

1.1 Supervised machine learning

Supervised learning, also known as supervised machine learning, is defined by the use of labelled datasets to train algorithms to classify data or predict an outcome. The model adjusts his weight as input data is fed into. Some methods that use supervised learning are neural networks, naïve bayes, linear regression, random forest and support vector machine (SVM).

1.2 Unsupervised machine learning

Unsupervised learning, also known as unsupervised machine learning uses unlabelled datasets to train machine learning algorithms. These types of algorithms can discover hidden patterns without the need for any kind of human intervention. Examples of algorithms that use this kind of method are neural networks, k-means clustering and probabilistic clustering methods.

1.3 Semi-supervised machine learning

This method provides a happy medium between the two methods previously presented. It uses a smaller labelled dataset to guide classification and feature extraction from a larger dataset, an unlabelled one. Semi-supervised learning algorithms can be applied in, for example, speech analysis, internet content classification and protein sequence classification.

2 Machine learning algorithms

There are also several numbers of machine learning algorithms currently used. The most common ones include: neural networks, linear regression, logistic regression, decision trees and random forest. Below lies a brief description of the subsets presented above.

2.1 Neural networks

These types of algorithms simulate the way the human brain works, mimicking the way that biological neurons signal to one another. Neural networks are good at recognizing patterns such as malware or intrusion detection in an Intrusion Detection System (IDS). They also play an important role in the fields of image recognition, speech recognition and image creation.

2.2 Linear regression

Linear regression can be used to predict the value of a variable based on the value of another variable and their linear relationship between them. This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. For example, this technique can be applied in a model to predict the value of real estates based in a historical data for the area.

2.3 Logistic regression

This method is a supervised machine learning algorithms, and it makes predictions for categorical response variables, such as “yes or no” answers to questions. It has various applications fields, like for example, classifying spam or quality control on a production line.

2.4 Decision trees

The decision tree is a supervised learning method that can be used for both predict numerical values (linear regression) and classifying data into categories.

It has a hierarchical – tree structure – which consists of a root node, branches, internal nodes and leaf nodes. It uses a branching sequence of linked decisions that will be represented in the hierarchical structure explained above. An advantage of these types of algorithms is that they are easy to validate or audit, unlike, for example, neural networks.

2.5 Random forests

This algorithm predicts the outcome – value or category – by combining the output of multiple decision trees. Like said above, this algorithm can be used to solve both regression or classification problems.

Part III
Neural networks

Neural networks, like presented in the chapter above, are a subset of machine learning. They are also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), and their structure were inspired by the human brain, from studies of the mechanisms for information processing in biological nervous systems.

In order to mimic a biological neural network, what we know as artificial neural networks (ANNs), a simple mathematical model of a single neuron was introduced in 1943 by the authors McCulloch and Pitts. The model takes the form indicated in *Figure 2* present below.

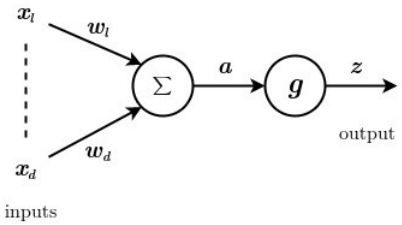

Figure 2: McCulloch-Pitts model

This model represents a non-linear function which creates weighted sums of the inputs x_1, x_d, x_n, given by $a = \Sigma_i w_i x_i$ and then transforms this sum using the non-linear activation function $g()$ to give a final out $z = g(a)$ put. In this representation the signal x_i is first multiplied by the parameter w_i known as *weight* – which is an analogy to the synaptic strength in a biological network – and then is added to all the other weighted inputs. The sums between the multiplication of inputs and weights creates the following equation.

(a) $a = \sum_{i=1}^{d} w_i x_i + w_0$, *where* w_0 *is a bias*

The bias helps to control the value at which the activation function will trigger. Some authors assume the parameter *bias* as a special case of a weight from an input whose value is constantly set to 1. To be accurate to the method presented above, the output of the z unit is then given by the operation of a non-linear function $g()$ with a, so that $z = g(a)$.

Part IV
Multilayer perceptron

By definition, a perceptron is an algorithm for supervised learning of binary classifiers. It is a type of linear classifier, i.e., a classification algorithm that makes predictions based on a sum between several inputs and the multiplication with their weights.

The multilayer perceptron (MLP) was developed to tackle a specific limitation of the definition presented above about the perceptron which is the linear mapping between inputs and outputs. An MLP support a non-linear mapping between these two, which has a direct impact in the complexity of the areas of learning and performance of the network.

This type of artificial neural network class is commonly referred to as neural networks, because of the layers of action that he takes. It consists of at least three layers: input layer, output layer and at least one hidden layer.

The input layer accepts input values without any kind of computation performance made. These nodes just pass on the information to the hidden layer. Hidden layers are an abstraction provided by any neural network. In this layer, all sorts of calculations are done by the values provided through the input layer. The result of the calculations are then transferred to the output layer which only brings up the information learned by the network.

1 Activation functions

Except for the input nodes, every other node is a neuron that uses a non-linear activation function. This function decides if a neuron should be activated or not by calculating the weighted sum and further adding bias to it. A few variants of this activation function are the following functions: Linear, Sigmodal, Rectified linear units (ReLU), Tahn and Softmax.

1.1 Sigmodal

The sigmoid function, also known as logistic activation function has a non-linear nature, and it's a typical choice for a feedforward neural network that expects and output range of 0 or 1, since the sigmoid function lies between these two values, so the result can be predicted 1 if the value is greater than 0.5 and 0 otherwise. It's represented by the following equation.

(a) $\phi(x) = \frac{1}{1 + e^{-x}}$

1.2 Rectified linear units (ReLU)

Since the first introduction of this activation function, the researches have rapidly adopted this method for their works due to superior training results in comparison to others functions like linear or hyperbolic tangent. It's mostly used and recommended for hidden layers due to the easily back-propagation of errors. It has the following straightforward equation.

(a) $\phi(x) = max(0, x)$

1.3 Hyperbolic tangent (Tahn)

This activation function is a mathematically shifted version from the sigmoid function, and they can be derived from each other. Once again, it's a function that is preferable to be used in hidden layers of a neural network. The output values lies between -1 and 1 which means that the results from the hidden layer comes really close to 0, hence helps the centring data by bringing mean close to 0. The following equation shows the hyperbolic tangent activation function.

(a) $f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1$

1.4 Softmax

Like the other functions presented, it has a non-linear nature, and it's usually used when trying to handle multiple classes, hence the use in the output layer of the classifier is recommended. It goes by the following equation.

$$\text{(a)} \quad \phi_x(x) = \frac{exp\,(x_i)}{\Sigma_j\, exp\,(x_j)}$$

In the above equation i represents the index of the output neuron, and j represents the indexes of all neurons in the group. When Softmax is the activation function, the output of a single neuron is dependent on the other output neurons.

Part V
Metrics for multi-class classification

In order to test any multi-class classifier there are metric that come in handy to: i) comparing the performance of two different models, ii) analysing the behaviour of the same model by change specific parameters.

In this brief chapter, the most popular and used metrics are going to be explained for a better understanding.

1 Confusion matrix

This metric is composed by a cross table that records the number of occurrences between two raters, the true classification and the predicted one.

It is typical a $N\,x\,N$ matrix used for evaluating the performance of a classification model, where N is the number of target classes. This method is used by two-class classification concepts: i) precision, ii) recall. In the next paragraph it will be explained a bit more about them.

2 Precision & Recall

Precision, by definition, is the fraction between True Positives and the sum of True Positives with False Positives. The term True Positive means all the elements that have been labelled positive by the model and are, in fact, positive, while False Positive are elements that have been equally labelled positive by the model, but are actually false. Below lies the equation referent to this metric.

(a) $$Precision = \frac{TP}{TP + FP}$$

This model tries to answer the question of what proportion of positive identifications was actually correct. In other words, it tells us how much we can trust the model when it predicts and individual Positive.

In the other hand, Recall measures the model's predictive accuracy for the positive class, which means, it tries to see what was proportion of actual positives was identified correctly. This metric can be represented by the following equation.

(b) $$Recall = \frac{TP}{TP + FN}$$

In order to explain, the Recall is the fraction of True Positives elements by the total number of positively classified units, which means, the sum between True Positives and False Negatives. The False Negative elements are the ones that were labelled as negative by the model, but are in fact positive.

3 Accuracy

Accuracy is one of the most popular metrics in multi-class classification, and it's directly computed from the confusion matrix. Mathematically, accuracy is defined as follows.

(a) $$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The formula consists in the division between the sum of True Positives and True Negatives and the sums between True Positives, True Negatives, False Positives and False Negatives. In a very simplistic way, accuracy is the probability that the model prediction is correct.

4 F1-Score

The goal of the F1-Score is to combine the Precision and Recall metrics into a single metric, working as the harmonic mean between this two. The formula it's a computed average of Precision and Recall as shown below.

(a) $$F\,1 - Score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Since F1-Score is an average between the two metrics presented above, it means that the F1-Score gives equal weight to Precision and Recall. If a model obtains a high F1-Score it means both Precision and Recall are high, the same goes for the lower approach, but if F1-Score is a medium value, that means that one of the Precision and Recall are high and the other one is low.

Part VI
Conclusion & Discussion

The universe of machine learning and their subsets are enormous, but this document only focus on a specific subset, neural networks. These type of networks can help us in the work that we are insight, which is the construction of an Intrusion Detection System (IDS), capable of identifying the most diverse anomalies and malware in a scenario of a Smart City.

With the explanation above presented in mind and with the information gathered and explained in this document, it's believed that with a good heterogeneous and balanced dataset it's possible to train a network capable of identifying an intrusion in a scenario like the one present above.

The performance of that network/model it will depend on the correct approach of a pre-processing data alongside with a good choice of the hyperparameters (e.g., activation functions, number of layers in the network, type of neurons) and classification metrics for the output result.

In conclusion, with the knowledge granted with this research it's possible to develop the ideal IDS for a hypothetical Smart City network.