International Journal of Progressive Sciences and Technologies (IJPSAT)
ISSN: 2509-0119.

Vol. 32 No. 1 April 2022, pp. 208-221

Classification Of Radar Targets Using Machine Learning

RANDRIANANDRASANA Marie Emile [1], RANDRIAMITANTSOA Paul Auguste[2], RANDRIAMITANTSOA Andry Auguste[3]

[1]Dept. of Telecommunication, Antsirabe Vankinankaratra High Education Institute, University of Antananarivo, Madagascar,
[2]Dept. of Telecommunication, High School Polytechnic of Antananarivo, University of Antananarivo, Madagascar,
[3]Dept. of Telecommunication, High School Polytechnic of Antananarivo, University of Antananarivo, Madagascar,

Abstract – This paper explores the use of machine learning for radar target classification. The techniques and algorithms used seek the best prediction for classification as well as the identification of certain characteristic properties of targets. A comprehensive and structured review of the application of machine learning based algorithms in radar signal processing such as Support Vector Machine (SVM), neural network is explored: (1) feature classification using VMS obtained from the test set. (2) classification of radar echoes using the neural network.

Keywords – Machine Learning, Radar, Gradient Descent, Support Vector Machine, Neural Network, Classification.

I. INTRODUCTION

Target classification is an important function in modern radar systems. Due to the recent success of using machine learning techniques for classification, there is much interest in applying similar techniques to classify radar returns. However, judicious implementation of machine learning classifiers is not always straightforward due to training data and computational requirements.

II. AUTOMATIC LEARNING

According to its inventor Arthur Samuel, machine learning consists in letting a machine learn without programming it explicitly.

Another American by the name of Tom Mitchell gave in 1998 a slightly more modern definition of machine learning by stating that a machine learns when its performance in a certain task improves with new experiences. [1]

To give a machine the ability to learn, learning methods are used that are heavily inspired by the way we humans learn to do things. These methods include:

- Supervised learning
- Unsupervised learning

A. Supervised learning

This learning consists in showing the machine some examples *(x, y)* and asking it to find the association that links *x* to *y*. [2]

In order to master supervised learning, one must understand and know the following four (4) concepts :

Corresponding Author: RANDRIANANDRASANA Marie Emile

- The DataSet
- The model and its parameters
- The cost function
- The learning algorithm.

A.1. **DataSet**

In supervised learning, we compile these examples (x,y) in an array that we call **DataSet**.

- The variable *there* is called **target**. It is the value we are trying to predict
- The variable *x is* called **features**. A factor influences the value of *y* and we usually have many features (x_1 , x_2 , x_3 ...) in our DataSet that we group in a matrix ***X***.

Table 01: A DataSet that groups examples of apartments with their price *and* some of their features

Target	***Features***		
y	x_1	x_2	x_3
Price	Surface area m^2	N rooms	Quality
€ 313, 000.000	124	3	1.5
€ 2, 384,000.000	339	5	2.5
€ 342, 000.000	179	3	2
€ 420, 000.000	186	3	2.25
€ 550 ,000.000	180	4	2.5
€ 490 ,000.000	82	2	1
€ 335, 000.000	125	2	2

A.2. **Model**

In supervised learning, we develop a model from the DataSet. It can be a **linear** model**,** or a **non-linear model.** The model is nothing else than a mathematical function. The coefficients of this function are the **parameters** of the model.

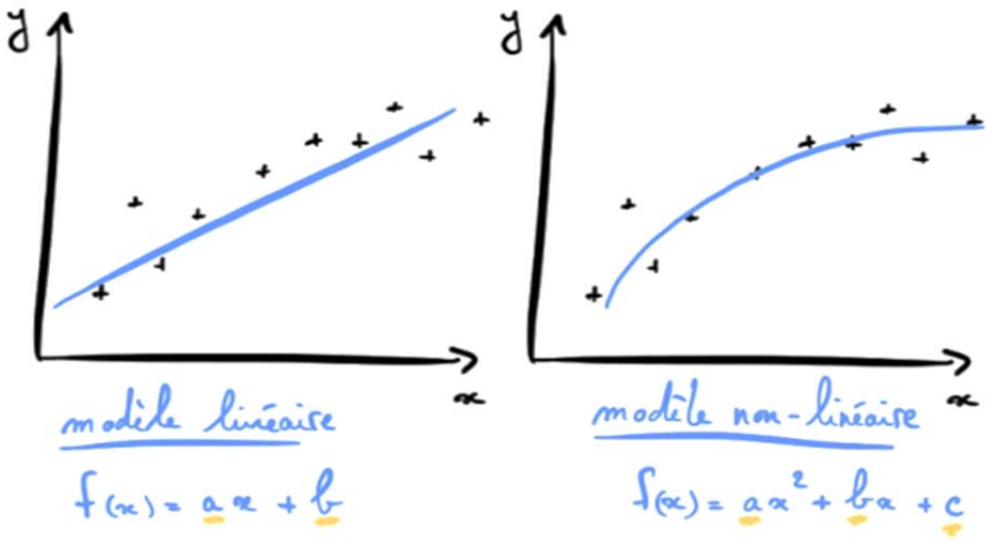

Fig 01: Model type

A.3 **Cost function**

The models we have return errors with respect to the DataSet. The set of all these errors forms the Cost Function (the most used is the root mean square of the errors.

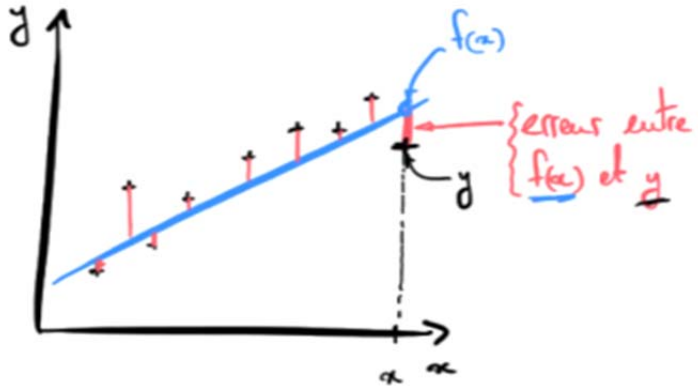

Fig02 : errors between f(x) and y

In machine learning, having a good model is a model that gives us small errors, i.e. a small cost function.

A.4 learning algorithm

The central objective of machine learning is to find the model parameters that minimize the cost function. The use of a learning algorithm is one of the methods used for this minimization. The most common example is the **Gradient Descent** algorithm.

With supervised learning we can develop models to solve two types of problems.

- Regression problems
- Classification problems

In the regression problem, we try to predict the value of a continuous variable, that is, a variable that can take an infinite number of values. In the classification problem, we try to classify objects in different classes, i.e., we try to predict the value of a discrete variable that takes only a finite number of values.

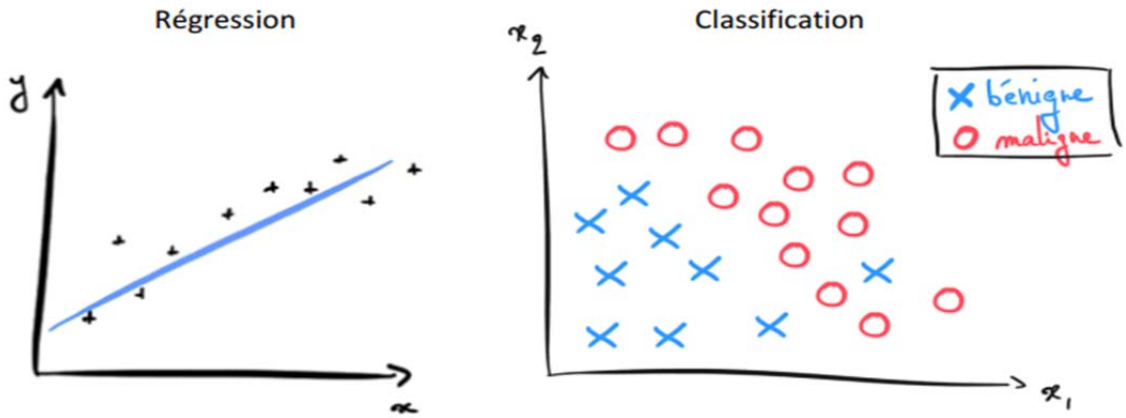

Fig 03 : Type of problems

B. Regression problems

B.1 Creating a linear model

Suppose we have a linear model $f(x) = ax + b$ where a and b are the parameters of the model, and we have m prediction example.

It is known that a good model gives small errors between its predictions $f(x)$ and the examples y of the Dataset.

For linear regression, we use the **Euclidean norm** to measure the errors between $f(x)$ and y. Here is the formula that expresses the error.

$$erreur = (f(x_i) - y_i)^2 \tag{1}$$

As each prediction is accompanied by an error, there are m errors. We define the **cost function** $J(a, b)$ as **the average** of all errors.

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m} erreur \tag{2}$$

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m} (f(x_i) - y_i)^2 \tag{3}$$

The cost function then becomes :

$$J(a,b) = \frac{1}{2m}\sum_{i=1}^{m} (ax_i + b - y_i)^2 \tag{4}$$

B.2 The Gradient Descent algorithm

The Gradient algorithm is a differentiable optimization algorithm. It is therefore intended to minimize a differentiable real function defined on a Euclidean space or, more generally, on a Hilbertian space. It is an iterative algorithm and therefore proceeds by successive improvements. At the current point, a displacement is made in the opposite direction of the gradient, in order to make the function decrease. The displacement along this direction is determined by the numerical technique known as linear search.

The gradient algorithm can then be used to find the minimum of the cost function *J(a, b)* starting from random coordinates ***a, b.*** Here is the step to follow:

1- Compute the slope of the cost function, i.e. the derivative of *J(a, b)*

2- Evolve a certain distance $\boldsymbol{\alpha}$ in the direction of the steepest slope. This results in changing the parameters **a** and **b.**

3- Repeat steps 1 and 2 until the minimum of *J(a, b)* is reached

$$\boldsymbol{a = a - \alpha \frac{\partial J(a,b)}{\partial a}} \quad \textbf{(5)}$$

$$\boldsymbol{b = b - \alpha \frac{\partial J(a,b)}{\partial b}} \quad \textbf{(6)}$$

At each iteration of this loop, the parameters *a and b* are updated by subtracting their own value from the value of the slope $\frac{\partial J(a,b)}{\partial \dots}$ multiplied by the distance to travel α.

This is called α the speed of learning.

In practice, the DataSet and the parameters are in matrix form, which greatly simplifies the calculations.

Let $\theta = \begin{pmatrix} a \\ b \end{pmatrix} \epsilon R^{n+1}$ which contains all the parameters for the model. X $\epsilon R^{m\,X\,n}$which contains all the features.

The linear model becomes: F(X)=X.θ where $\theta = \begin{pmatrix} a \\ b \end{pmatrix}$

The cost function: $\boldsymbol{J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (F(X) - y)^2}$

The Gradient and the algorithm : $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{2m} X^T \sum_{i=1}^{m} (F(X) - y)$

In the classification problem, which consists for example in classifying an email as 'spam' or 'not spam'. In this kind of problem, the dataset containing a target variable *y* can take only two values:

- y = 0 if a mail is not spam
- y = 1 if a mail is spam

We also say that we have 2 classes, it is a binary classification.

For these problems, a decision boundary is added to the model to classify an email in **class 0** or **class 1.**

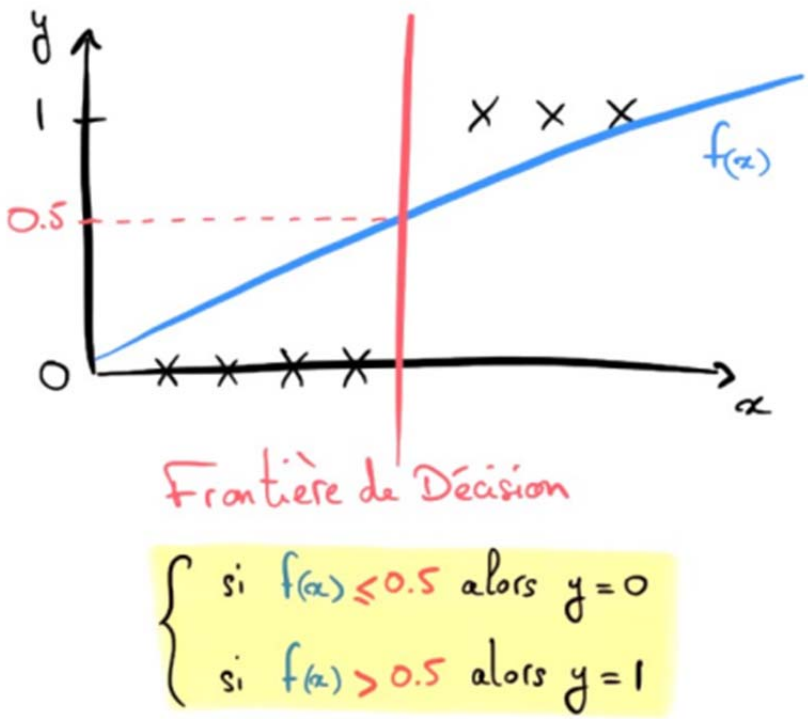

Fig 04: Classification problem

C. Classification problems

C.1 The Logistic regression model

A linear model $F(X) = X.\theta$ is not suitable for the binary classification problem. We then develop a new function for binary classification problems, it is the **logistic function** (also called **sigmoid function** or simply **sigma σ**); a function which has the particularity to be always between 0 and 1.

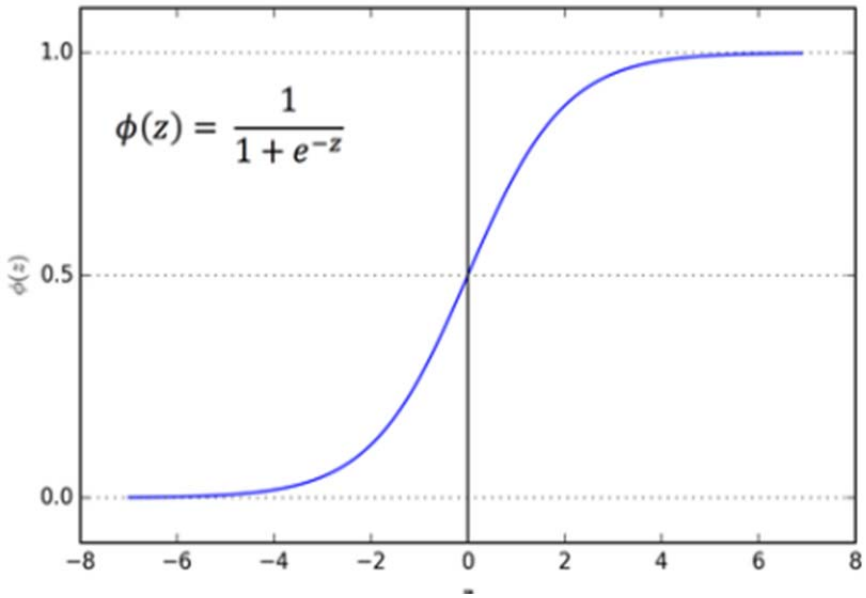

Fig 05 : representation of the sigma function

By introducing the logistic function on a dataset *(X, y)*. We pass the matrix product $X.\theta$ and we obtain the logistic model :

$$\sigma(x, \theta) = \frac{1}{1 + e^{X\theta}} \tag{7}$$

From this model, it is possible to define a decision boundary.

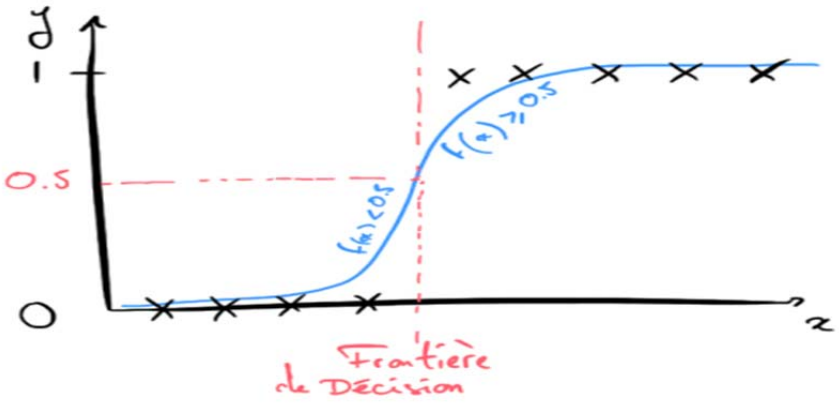

Fig 06 : decision boundary

For linear regression, the function $J(\theta)$ gives a convex curve (which has a unique minima). This is what makes the Gradient Descent algorithm work.

However, using this function for the Logistic model will not give a convex curve (non-convex function) and the Gradient Descent algorithm will not work.

A new cost function must be developed specifically for logistic regression. The **logarithm** function is made to transform the sigma function into a convex function by separating the cases where *y = 1* :

Cost function in cases where *y = 1*

$$\boldsymbol{J(\theta) = -log\big(\sigma(X \cdot \theta)\big)} \tag{8}$$

If the model predicts $\sigma(x) = 0$ while *y = 1*, we must penalize the machine with a large error (a large cost). The logarithm function allows to draw this curve with a convex property, which will push the Gradient Descent to find the parameters θ for a cost that tends to 0.

Cost function in cases where *y = 0*

$$J(\theta) = -log\big(1 - \sigma(X \cdot \theta)\big) \tag{9}$$

If the model predicts $\sigma(x) = 1$ while *y = 0*, we must penalize the machine by a large error (a large cost). This time **- log (1 - 0)** gives the same curve, inverted on the vertical axis.

To write the cost function in a single equation, we must use the trick of separating the cases *y = 0 and y = 1*

$$J(\theta) = \frac{-1}{m}\sum y\,.\,log\big(\sigma(X\theta)\big) + (1-y).\,log(1-\ \sigma(X\theta) \tag{10}$$

In the case where *y = 0*, there remains only :

$$J(\theta) = \frac{-1}{m}\sum log(1-\ \sigma(X\theta) \tag{11}$$

In the case where *y = 1*,

$$J(\theta) = \frac{-1}{m}\sum log\big(\sigma(X\theta)\big) \tag{12}$$

C. 2 radient Descent for logistic regression

The Gradient Descent algorithm is applied in exactly the same way as for linear regression. In addition, the derivative of the cost function is the same too:

Gradient: $\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}\Sigma\big(\sigma\big(X\cdot\theta\big) - y\big)\cdot X$

Gradient Descent: $\theta = \theta - \alpha\cdot\frac{\partial J(\theta)}{\partial \theta}$

C. 3 The Nearest Neighbour algorithm

It is an algorithm that allows to solve multi-class classification problems in a simple and very efficient way. It is enough to find in the DataSet the closest value with respect to the conditions in which we are looking.

- **K-Neareset Neighbour (K-NN)**

The k-nearest neighbors or K-NN classifier is one of the simplest classification algorithms. An example is classified by majority vote of its *k* "neighbors" (by distance measure), The most commonly used distance operator is the **Euclidean distance,** however, depending on the problem, one can still use ***Hamming*** distances**,** ***Mahalanobis*** distances**,** *etc.*

The choice of *k* is very important for the classification, and we refrain from choosing it for even values to avoid cases of equality.

C.4 Naive Bayesian Classifier

The naive Bayesian classification is based on the hypothesis that the attributes are strongly independent. It is based on the BAYES theorem which only applies under this assumption. [2] [9]

Bayes' theorem is given by : $\boldsymbol{P(x|y)} = \frac{\boldsymbol{P(y|x)P(x)}}{\boldsymbol{P(y)}}$ with $\boldsymbol{P(x|y)}$ is the conditional probability of an event *x* knowing that another event *y* of non-zero probability has occurred.

In the case of a classification, we pose *H* the hypothesis that a "vector of attributes *X* (representing an object) belongs to a class C", and we suppose that we are trying to estimate the probability *P(H|X), i*.e. the probability that the hypothesis *H* is true, considering X.

Thus, if we have a new example x = (x_1 , x_2 , x_3 ,... x_n) whose class we want to find, we will look for the maximum probability of belonging to this class :

$$\boldsymbol{P(x)^* = arg\,max\, P(x_1, x_2, x_3 \ldots x_n|H) * P(H)} \quad \textbf{(13)}$$

C.5 Neural Networks

Neural networks are much more complex models than any other machine learning models in the sense that they represent mathematical functions with millions of coefficients (the parameters). With such power, it is possible to train the machine on much more advanced tasks [7] :

- Object recognition and facial recognition
- Sentiment analysis
- Natural language analysis
- Artistic creation
- Etc

However, developing such a complex function comes at a cost. To achieve this, it is often necessary to provide :

- A much larger dataset (millions of data)
- Longer learning time
- More computing power

In the field of machine learning, researchers have developed variants of the Gradient Descent as well as other techniques to compute derivatives on millions of data sets more quickly. Among these solutions are: [3]

- Mini-Batch Gradient Descent: technique where the dataset is fragmented into small batches to simplify the calculation of the gradient at each iteration.
- Batch normalization: to scale all the input and output variables internal to the neural network to avoid having extreme gradient calculations.
- Distributed deep learning: using the cloud to divide the work and entrust it to several machines.

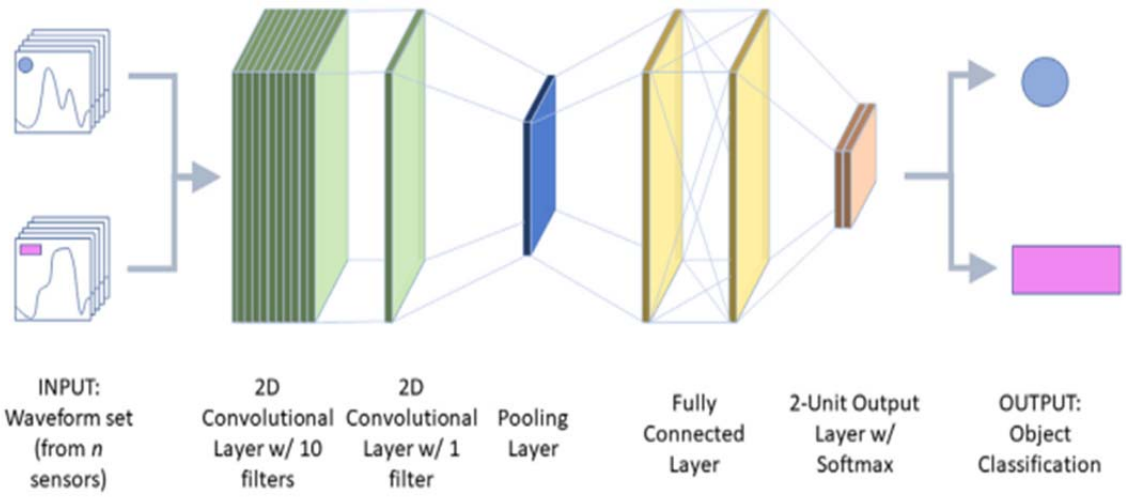

Fig 07 : Structure of the convoluted neural network

C.6 Support vector machine

Support vector machines or wide margin separators (called SVM for *Support Vector Machine*). SVMs can be applied to both linearly separable and non-separable problems. [8]

The main idea is to reconsider the problem in a higher dimensional space, possibly an infinite dimensional one. In this new space, it is then likely that there is a linear separating hyperplane. If this is the case, the SVMs search among the infinite number of separating hyperplanes for the one that maximizes the margin between the classes.

Applied to the input vector' x a nonlinear transformationΦ□□□in order to describe the data in another space. The input spaceΦ $(x$□□is called the re-description space. We then look for the optimal separating hyperplane, of equation $h(x) = \alpha\phi(x) + \beta$ in the redescription space.

We look for the equations of the parallel hyperplanes which pass through the support vectors, i.e. the attributes closest to the interclass boundary. We deduce the equation of the optimal hyperplane, equidistant from these hyperplanes.

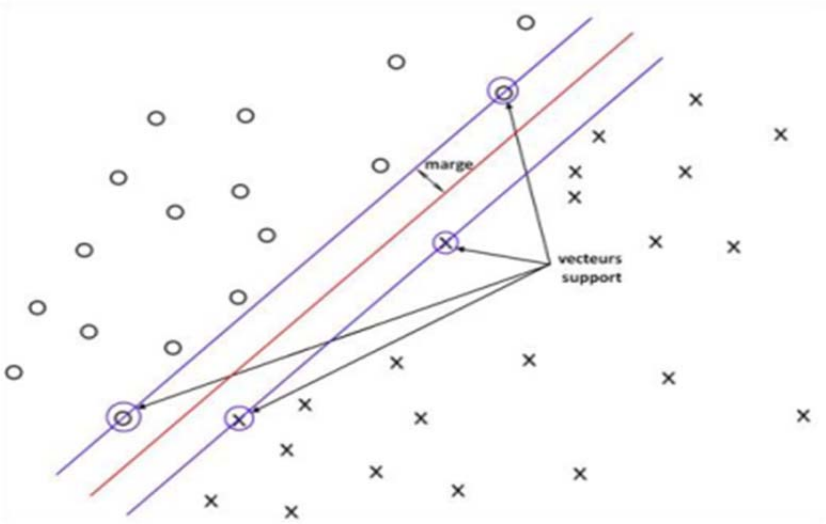

Fig 08: SVM principle

D. Unsupervised learning

From a certain point of view, supervised learning consists in teaching the machine things we already know, since we build in advance a dataset containing *X* questions and *Y* answers.

In unsupervised learning, we have a dataset (*x*) with no value (*y*), and the machine learns to recognize structures in the data (*x)* that we show it.

We can thus group data in clusters (this is clustering), detect anomalies, or reduce the dimension of very rich data by compiling the dimensions together. [1] [2]

K-Mean Clustering Algorithm

It is the most popular algorithm for clustering problems (grouping data according to their common structure). [1] [2]

The algorithm works in two steps repeated in a loop. Starting with the random placement of a number **K** of **centers** in the point cloud.

III. Classification Of Radar Targets Using Machine Learning

This is an example that shows how to classify radar returns with machine learning approaches. The machine learning approach uses wavelet scattering feature extraction coupled with a support vector machine. In addition, two learning approaches are illustrated: transfer learning using SqueezeNet and a neural network.

By classifying the radar echoes of a cylinder and a cone,

A. Summary of RCS

In order to create the synthesized data for training the learning algorithm, the RCS model of a cylinder and cone with a radius of 1 meter and a height of 10 meters was simulated. The operating frequency of the radar is 850 Mhz.

The pattern can then be applied to a backscatter radar target to simulate returns at different aspect angles.

The following graph shows how to simulate 100 returns of the cylinder over time. It is assumed that the cylinder undergoes a movement that causes small vibrations around the bore, therefore the aspect angle changes from sample to sample.

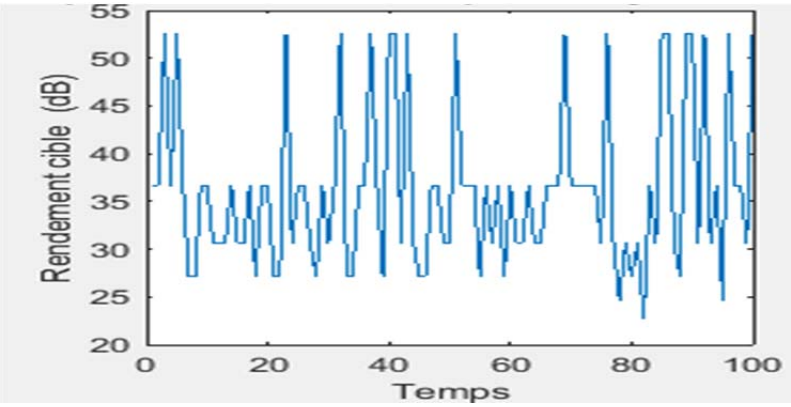

Fig 09 : return of the target for the cylinder

To create the training set, the process is repeated for 5 arbitrarily selected cylinder rays. In addition, for each ray, 10 motion profiles are simulated by varying the angle of incidence along 10 randomly generated sinusoidal layers around the viewing axis. There are 701 samples in each motion profile, so there are 701 per 50 samples. The process is repeated for the cylinder target, resulting in a 701 x 100 matrix of training data with 50 cylindrical and 50 conical profiles. In the test set, we will use 25 cylindrical profiles and 25 conical profiles to create a 701 by 50 training set.

The following plot shows the return for one of the motion profiles for each shape. The plots show how the values change over time for the incident azimuth angles and target returns.

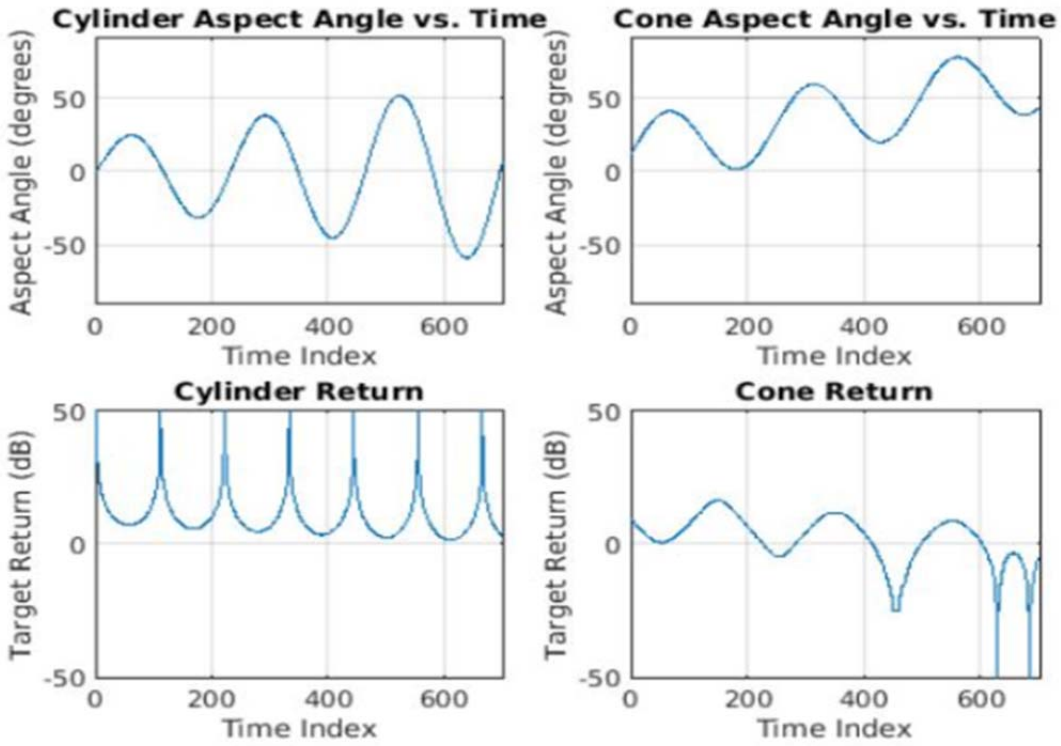

Fig 10 : return of a profile

To improve the matching performance of learning algorithms, learning algorithms often work on extracted features rather than the original signals. The features facilitate the algorithm classification to discriminate returns from different targets. In addition, the features are often smaller in size compared to the original signal so that it requires less computational resources to learn.

There are several ways to extract features for this type of data set. To get the right feature, it is often useful to look at a time-frequency view of the data where the frequency due to motion varies from one radar pulse to another. The time-frequency signature of the signal can be derived either by Fourier transform (the spectrogram) or by wavelet transforms.

The following graphs show the wavelet packet signatures for the cone and cylinder. These signatures give an idea that the learning algorithms will be able to distinguish between the two. Indeed, there is a separation of the frequency content in time between the two signatures.

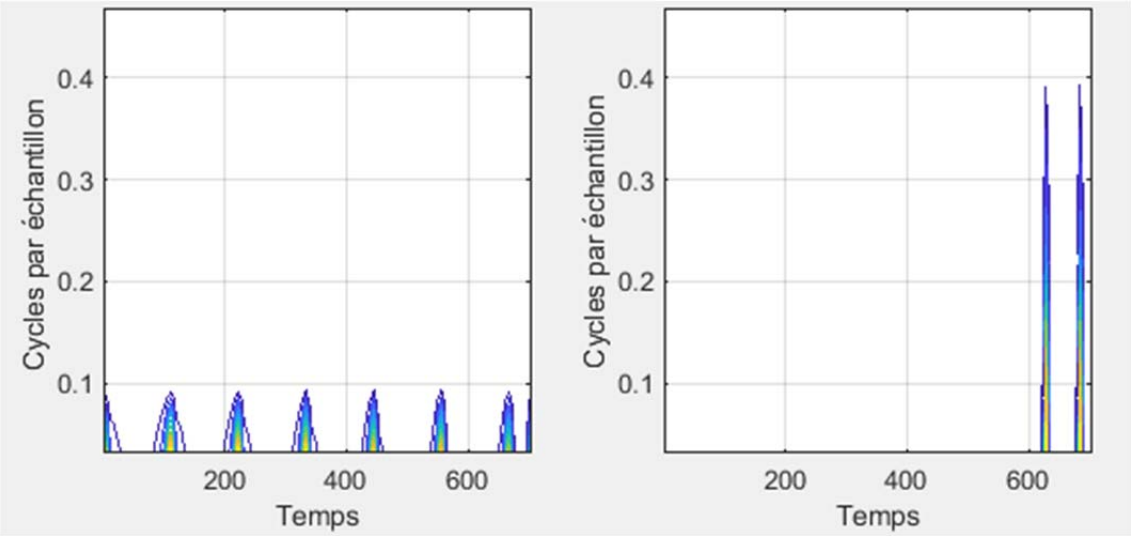

Fig 11 : Wavelet package for cylinder and cone

In the wavelet diffusion feature extractor, data are propagated through a series of wavelet transforms, nonlinearities, and averaging to produce low variance representations of the time series. Wavelet time diffusion produces signal representations that are insensitive to shifts in the input signal without sacrificing class discriminability.

The key parameters to specify in a wavelet time-scattering network are the scale of the time invariant, the number of wavelet transforms, and the number of wavelets per octave in each of the wavelet filter banks. In many applications, cascading two filter banks is sufficient to achieve good performance. In our example, we build a wavelet time-scattering network with the two filter banks: 4 wavelets per octave in the first filter bank and 2 wavelets per octave in the second filter bank. The invariance scale is set to 701 samples, the length of the data.

Next, we obtain the diffusion transforms of the training and test sets.

Using the trained VMS, the scattering characteristics obtained from the test set were ranked.

B. Transfer learning with CNN

SqueezeNet is a deep convolution neural network trained for images in 1000 classes, as used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In our simulation, we reuse the pre-trained SqueezeNet to classify radar echoes belonging to one of two classes. [6]

SqueezeNet consists of 68 layers. Like all DCNNs, SqueezeNet cascades convolutional operations followed by nonlinearities and pooling, or averaging. SqueezeNet expects an image input of size 227 by 227 by 3.

SqueezeNet is also designed to discriminate differences between images and classify the results. Therefore, in order to use it to classify radar echoes, we need to transform the 1-D radar echo time series into an image. A common way to do this is to use the time-frequency representation (TFR). There are a number of choices for a time-frequency representation of a signal and which one is most appropriate depends on the characteristics of the signal. To determine which TFR may be appropriate for this problem, we randomly selected and plotted a few radar returns from each class.

It is then obvious that the radar echoes shown above are characterized by slow variable variations punctuated by strong transient decreases as described above. A wavelet transform is ideally suited to the parsimonious representation of such signals. The wavelets shrink to localize transients with high temporal resolution and stretch to capture a slowly varying signal structure.

IV. Conclusion

These examples present a workflow for performing radar target classification using machine learning techniques. Although synthesized data from training and testing was used, it can be easily extended to take into account real radar returns.

Due to the characteristics of the signal, techniques have been used for both learning approaches. Still, it is possible to improve the performance of the classifier by increasing the quality and quantity of the training data. In addition, the feature extraction process can be improved to further distinguish the features of each target in the classification algorithm.

REFERENCES

[1] Guillaume St-Cirgue " Learn machine learning in a week ", 2019

[2] Frédéric SUR "Introduction of machine learning", 2021 - 2022

[3] Aihua Wood, Ryan Wood, Matthew Charnley "Through-the-wall radar derection using machine learning," 2020

[4] Jean-Marie Alder "Low altitude obstacle detection for UAVs using machine learning", 2020

[5] Antoine Cornuéjols - Laurent Miclet " Artificial learning - Concepts and algorithms ",

[6] Rudolph Russell "Machine learning step-by-guide to implement machine learning algorithms with python" 2018

[7] Alessandro Davoli " Machine learning and Deep Learning Techniques for colocated MIMO Radars: A Tutorial Overview, January 2021

[8] Florian Kraus, Nicolas Sheiner, Werner Ritter, Klaus Dietmayer "Using Machine Learning to Detect Ghost Images in Automative Radar," July 2020.

[9] J.Francis Roy "Machine learning with generalization guarantees using disagreement-maximizing ensemble methods," 2018

[10] « Matlab expo 2018 » Machine learning for radar & EW

[11] Guy Ardon, Or Simko, Akiva Novoselsky «Aerial Radar Target Classification using Artificial Neural Network » Avril 2020