Obtaining Anti-Missile Decoy Launch Solution From a Ship Using Machine Learning Techniques

Ramón Touza[1*], Javier Martínez[2], María Álvarez[3], Javier Roca[2]

[1] Spanish Naval Academy, Pontevedra (Spain)
[2] Vigo University, Pontevedra (Spain)
[3] Defense University Center, Pontevedra (Spain)

Received 11 November 2020 | Accepted 2 March 2021 | Published 3 November 2021

Abstract

One of the most dangerous situations a warship may face is a missile attack launched from other ships, aircrafts, submarines or land. In addition, given the current scenario, it is not ruled out that a terrorist group may acquire missiles and use them against ships operating close to the coast, which increases their vulnerability due to the limited reaction time. One of the means the ship has for its defense are decoys, designed to deceive the enemy missile. However, for their use to be effective it is necessary to obtain, in a quick way, a valid launching solution. The purpose of this article is to design a methodology to solve the problem of decoy launching and to provide the ship immediately with the necessary data to make the firing decision. To solve the problem machine learning models (neural networks and support vector machines) and a set of training data obtained in simulations will be used. The performance measures obtained with the implementation of multilayer perceptron models allow the replacement of the current procedures based on tables and launching rules with machine learning algorithms that are more flexible and adaptable to a larger number of scenarios.

Keywords

Decoys, Machine Learning, Missile, Multilayer Perceptron, Support Vector Machine.

DOI: 10.9781/ijimai.2021.11.001

I. Introduction

The missile is one of the most dangerous threats a warship can face (see Fig. 1). Currently, about eighty countries have anti-ship missiles in their arsenals, which can be launched from aircraft, ships, submarines or from a coastal battery.

Fig. 1. USS Stark after the impact of two missiles launched from an Iranian aircraft in the Persian Gulf (Source: https://www.history.navy.mil).

* Corresponding author.
E-mail address: rtougil@fn.mde.es

Furthermore, given the current international scenario with an increase in global terrorism, it cannot be ruled out that anti-ship missiles may fall into the hands of some terrorist group and given that naval operations are more often conducted in coastal waters near the coast, this threat takes on special significance, as ships are exposed to attacks from land and with reduced reaction times.

In parallel with the change in the global threat, governments have increased their control over military forces deployed in areas of operation by dictating increasingly restrictive rules of engagement (ROE's) [1] and although every ship commander has an inherent right of self-defense, if an escalation of tension in the conflict zone is not desired, the use of force must always be proportional. Moreover, the need to avoid friendly confrontations makes it even more difficult to use the ship's weapons in combating a possible missile attack.

In addition to the foregoing, the Anti Surface Missile Defense (ASMD) is, from the tactical point of view, a complex action in which decisions must be taken in seconds, with no margin for error, and in which all means of defense may be employed: hardkill (long, medium and short-range missiles, naval guns and small arms) and softkill (passive and active electronic countermeasures and decoys).

The decoys are passive elements, which can be considered the last layer of defence of the ship. As shown in Fig. 2, the intention is to deceive the missile and focus its guidance system on a false target, this process, in naval tactical terminology, is known as seduction.

Another advantage of decoys is that their use does not involve the lethal use of force, which is always permitted by the rules of engagement.

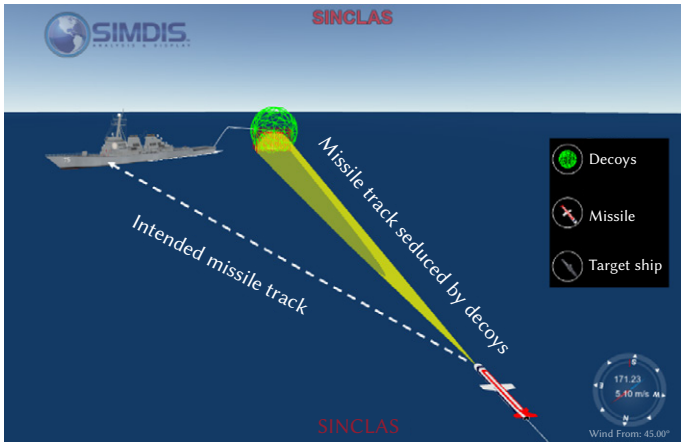

Fig. 2. Seducing a missile with decoys.

However, its use is not simple, to be effective and as shown in Fig. 3, the ship must react immediately, choose one of the 4 fixed launchers that are located in different positions and also must consider altering course to port or starboard to improve efficiency.

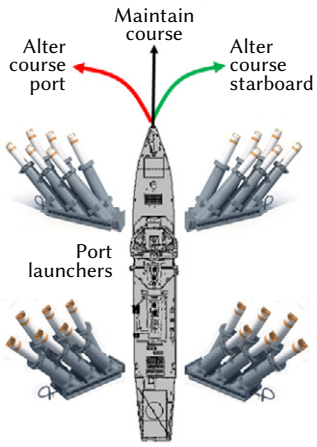

Fig. 3. Typical ship launchers configuration and possible maneuvers.

The launcher/alter course combination is what is known as the launch solution and therefore, for a scenario there are 12 possible solutions (4 launchers x 3 possible ship manoeuvres).

Currently, ships have implemented a series of pre-planned reactions, which they will use in the event of a missile attack. However, these reactions can only cover a limited number of scenarios.

Applying current technology of artificial intelligence, in particular machine learning techniques, it is possible to improve the accuracy of the reactions obtaining solutions for any scenario.

Moreover, for the same scenario there can be more than one launch solution, so within this last set it will be necessary to determine which is the best reaction to use. As an example, for two valid reactions, one may present a much greater miss distance of the missile from the ship than the other. In short, the feasible solutions for the same scenario can be ordered according to a series of criteria, allowing the ship to make a better decision.

Finally, as regards current anti-ship missiles, we can classify them, according to their guidance system, as: electro-optical (EO), infrared (IR), radio frequency (RF) or dual (combination of the above) and therefore, to defend against them, different types of decoys are used: flares for EO and IR guidance missiles, chaff for RF guidance missiles or a combination of the above for dual guidance missiles.

In this paper we will focus on the chaff launch solution for seducing radio frequency guidance missiles, although the methodology outlined here is applicable to any type of missile.

II. Objectives

The aim of this work is to develop a machine learning (ML) model which, trained from data from a number of simulations, obtains for any scenario what reactions of the ship allow to seduce the missile.

In addition, the ML model must have other data outputs: the expected missile miss distance, the time the ship is being tracked by the missile before it is focused on the decoy, and the probability of success of each possible launch solution. All these data can feed into a multi-criteria decision layer to obtain the best solution to the problem.

The basic architecture of the process would be as shown in Fig. 4, where a large number of scenarios would be simulated in the laboratory, to obtain a dataset with which to train a machine learning model, to be subsequently implemented on the ship and to obtain the best launch solution.

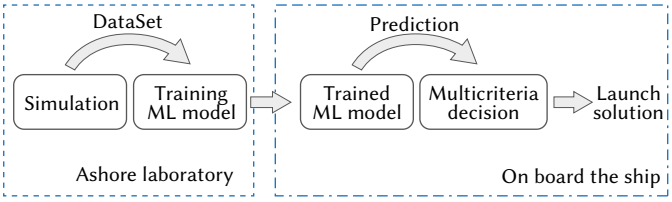

Fig. 4. Laboratory-on board process.

III. Simulation

In a real action against an attacking missile, the ship available time may not reach one minute, so the decoy launching solution must be pre-planned and there must also be a solution for all or most of the possible scenarios in which the ship may find itself.

The possible launch solutions are obtained by numerical simulation techniques before going to sea and are implemented on the ship by solution tables. This simulation is based on the interaction of three models: ship, decoy and missile, which interact in a scenario given by a wind, a ship speed and a distance and bearing of the attacking missile.

Each of the models involved in the simulation are characterized by a series of parameters or characteristics that will affect the launch solution.

The ship model is mainly characterized by its dimensions, turning circle maneuver, position of the decoy launchers and the radar cross section (RCS). The RCS is a random measurement that fluctuates rapidly over time and whose value determines the size of the echo that the ship presents on the radar of the attacking missile. Therefore a low RCS will make the vessel less visible to the missile seeker radar and will make it easier for the decoys to attract it.

The missile model is characterized by its speed and height of flight and certain parameters of its seeker radar.

On the other hand, the decoy model will be characterized by its deployment data: distance, height, deployment time, drop speed, cloud diameter and by its radar cross section. The decoy RCS is also a random variable and is an extremely important data, since it will have to have a sufficiently high value, in relation to the RCS of the ship to deceive the missile.

In addition to the three models mentioned, each of the possible simulation scenarios is given by the ship speed, the bearing (direction)

and distance of the attacking missile and the wind (direction and intensity). The wind parameter is especially relevant since it will be responsible for dragging the cloud formed by the decoy away from the ship, which should foil the missile from its intended target.

The three models interact in the simulation in order to calculate, in each scenario, the launch solution that allows the seduction of the enemy missile. The seduction mechanism consists of deploying a cloud or clouds of decoys with sufficient RCS so that, due to the effect of the wind and the speed of the ship, the seeker radar resolution cell of the attacking missile will be centered on the cloud of decoys and away from the ship. This effect is called the centroid effect[2], which can be summarized in the phases shown in Fig. 5 [3]. Initially, in Phase A, the missile has the ship inside its seeker radar resolution cell and the ship reacts with a decoys launch. In Phase B, due to the effect of the wind, the decoys separate from the ship and the missile centers its resolution cell in the center of all the RCS (ship and decoys) contained in the cell. Finally, in Phase C the decoy continues to move away from the ship and if it has enough RCS, the transfer of the tracking from ship to decoy takes place.

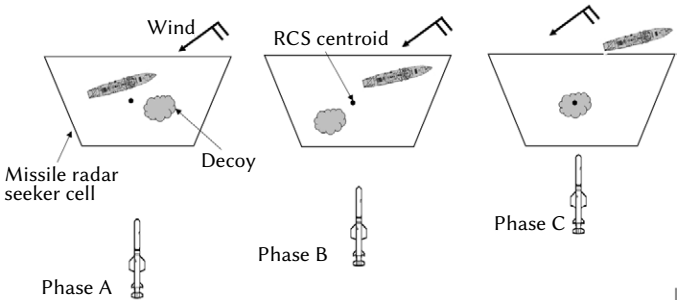

Fig. 5. Centroid effect.

The missile, ship and decoy models interact in the simulator as described for the centroid effect and it allows us to determine for each scenario which of the twelve ship possible reactions (combination of launcher and change of course) are effective in seducing the missile, i.e. success/failure. In addition, other data are obtained from the simulation, such as the minimum miss distance of the missile from the ship and the percentage of time that the missile has centered the ship in the seeker radar resolution cell before transferring the tracking to the decoy.

It should be noted that, as it is a stochastic simulator, a certain number of runs must be made in order to draw conclusions and therefore, for each possible solution, a probability of success can also be obtained.

In this work, a series of scenarios have been configured in the simulator with the data shown in Table I, giving a total of 186,624 scenarios. As there are 12 possible solutions per scenario, a total of 2,239,488 instances are obtained, of which 30 runs have been made. This provides a sufficient volume of data for training, validation and testing of any machine learning model.

TABLE I. Simulation Scenario Data

Scenario parameter	Values
Ship speed (S_{ship})	10, 15, 20 knots
Wind speed (S_{wind})	0, 10, 20 ,30 knots
Wind bearing (B_{wind})	[0°-355°] step 5°
Missile distance (D_{misil})	5000, 10000,15000 yards
Missile bearing (B_{misil})	[0°-355°] step 5°

Fig. 6 shows how the wind and missile bearing is measured.

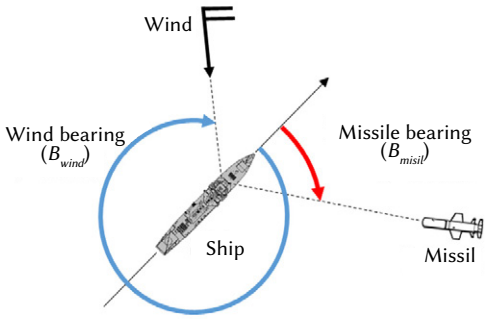

Fig. 6. Wind and missile bearing.

Before focusing on the types of ML models that can solve the problem, it is necessary to clarify the available dataset obtained from the simulations. The ML model should have as inputs the parameters of the scenario in which the vessel is, the available launchers (4 launchers in our case) and the possible alter course. Thus, the input data will be as shown in Table II.

TABLE II. Input Data

Data	Units
Ship speed (S_{ship})	knots
Wind speed (S_{wind})	knots
Wind relative bearing (B_{wind})	degrees
Missile distance (D_{misil})	yards
Missile relative bearing (B_{misil})	degrees
Launcher (L)	1, 2, 3, 4 (binary coded)
Alter course (A)	Port, 0°, starboard (binary coded)

From the input data, for each scenario and for each possible launch solution, we will have the following Target data (see Table III).

TABLE III. Target Data

DATA	UNITS
Solution	1 – success (no impact on ship); 0-fail (impact on ship)
Miss distance	Yards
Time in cell	Total engagement percentage
Probability of success (Note)	1-Probability of success >0.8 0- Probability of success ≤0.8

Note: 0.8 is taken as the probability of success since it corresponds to the turning point of the probability/frequency histogram and encompasses 87% of the total solutions.

IV. Machine Learning Models

As mentioned above, the aim of this work is to build a machine learning model that, trained from the data obtained from the simulation, can be implemented in a ship's combat system so that it will provide the necessary outputs to determine the best decoy launching solution.

Since we have four types of target data, it was decided to build four different ML models based on supervised learning: two binary classification models for solution and probability of success, and two regression models for miss distance and time in cell.

Initially, two types of machine learning techniques were chosen as possible candidates to solve the problem: Support Vector Machine (SVM) and Multi Layer Perceptron (MLP). Thus, this study tries to determine which is the optimal technique or which combination of both techniques is the optimal one to solve the problem.

A. Multilayer Perceptron (MLP) Model.

The term neural network has its origin in attempts to find mathematical representations of information processing in biological systems. It is a powerful structure that allows the creation of non-linear predictive models and is used in supervised and unsupervised learning[4]. In the case of supervised learning it allows to build binary and multiclass regression and classification networks [5]–[7].

The basic structure of a neural network is the neuron (see Fig. 7), the input variables (x_i) are connected to the neuron through weighted connections (w_i) that emulate dendrites, while the sum (Σ), the bias (b) and the activation function (h) play the role of the cell body and the propagation of the output is analogous to the axon in a biological neuron. The behavior of the neural network is defined by the shape of the connections of its neurons or nodes and by the values of the weights of these connections. These weights are automatically adjusted during training according to a learning algorithm, until the network carries out the desired task correctly [8].

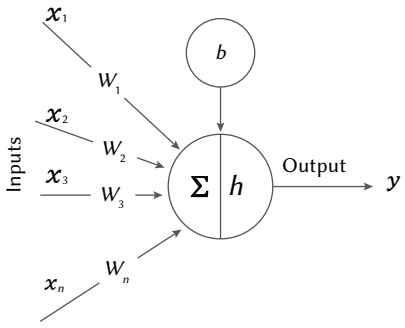

Fig. 7. Neuron model.

Multilayer perception (MLP) or multilayer network with feed-forward architecture, without feedback, is a kind of neural network, which has an architecture with a finite number of layers and neurons. In this structure three different types of layers are distinguished: the input layer, the hidden layers and the output layer, the latter having as many outputs as targets to be predicted. Generally, all the neurons in one layer are connected to all the neurons in the next layer, which is known as total connectivity or fully connected network.

The MLP characterizes the relationship between input and output layers, which is parameterized by the weights. This relationship is obtained by propagating the values of the input variables forward. To do this, each neuron in the network processes the information received by its inputs and produces a response or activation that propagates, through the corresponding connections, to the neurons in the next layer.

If we have an MLP with D inputs, and there are L layers with n_l neurons each one, the first $L-1$ layers will be the hidden layers and then there is the output layer, with so many outputs K as dimensions have the target to predict. The output of the neuron i of the layer l we will denote it by z_i^l. From one layer to the next, the output of the neuron i is weighted by a weight w_{ij}^l where l is the layer and j is the neuron of the layer l (see Fig. 8).

The weighted sum of the weights of a neuron input is called activation. Thus, for the neuron j of the layer l, the activation a_j^l is given by:

$$a_j^l = \sum_{i=0}^{n_{l-1}} w_{ji}^l \, z_i^{l-1}$$

This input is transformed using an activation function $h^{(l)}(\cdot)$ to generate the output z_j^l to the next layer:

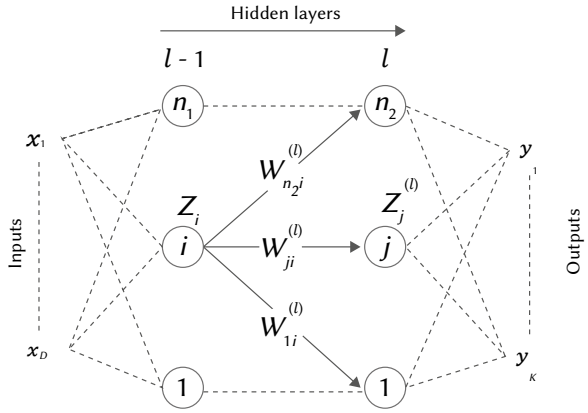

Fig. 8. Nomenclature used in the mathematical definition of neural networks.

$$z_j^l = h^{(l)}\left(a_j^{(l)}\right) = h^{(l)}\left(\sum_{i=0}^{n_{l-1}} w_{ji}^l \, z_i^{l-1}\right)$$

The activation functions for the output layer in our problem have been chosen according to the type of prediction. Thus, the softmax function was used for binary classification and the identity or purelin function for regression. For the hidden layers, the tansig function was used in all cases.

Now, from the training dataset given by some entries $\{\boldsymbol{x}_n\}$ and a target $\{\boldsymbol{t}_n\}$, the training objective is to find a set of weights $\boldsymbol{w}$ that minimizes the error function that we will denote $J_n(\boldsymbol{w})$ which measures the error associated with the training sample. The sum of squares function, below is a common choice:

$$J_n(\boldsymbol{w}) = \sum_{n=1}^{D} \|\boldsymbol{y}(\boldsymbol{x_n}, \boldsymbol{w}) - \boldsymbol{t_n}\|^2$$

To solve this problem, heuristic methods based on the descent of the gradient called backpropagation are used. The steps to carry out this algorithm are the following:

Step#1. Apply the inputs $\boldsymbol{x}_n$ to the MLP, propagate forward, calculate the outputs $\boldsymbol{y_k}$ and errors in the output layer $\boldsymbol{\delta}_k = \boldsymbol{y_k} - \boldsymbol{t_k}$

Step#2. Errors are propagated backwards, calculating for all hidden layers. So that, the error of a certain layer, is calculated on the basis of the error of the following layer:

$$\delta_j = h'(a_j) \sum_k w_{kj} \, \delta_k$$

Where the term δ_j is called error and must be calculated for each network node.

Step#3. Evaluation of error-function. Derivatives are calculated as:

$$\frac{\partial J_n}{\partial w_{ji}} = \delta_j z_i$$

Step#4. From the derivatives, bayesian regularization backpropagation algorithm is applied that progressively improves the weights of the network [9]–[12].

An important aspect to consider when designing a multilayer perceptron is the number of layers and neurons per layer. More than one hidden layer speeds up the training process, especially in very heavy problems. On the other hand, the more neurons per layer, the better the adaptability of the model to the problem. However, an increase in the number of layers and neurons can lead to overfitting and poor generalization [13], [14].

For this study, we have chosen to start with a single hidden layer network, and increase the number of layers while decreasing

the error. In the same way, the number of nodes or neurons in the hidden layers has been increased until the error is stabilized. In the case of configurations with more than one hidden layer, the pyramid technique has been used, which consists of decreasing the number of nodes from the input layer to the output layer [15].

Therefore, for the problem of this study, first of all, it is necessary to determine which the optimal configuration of layers and neurons per layer is for each of the four target data (solution, miss distance, time in cell and probability of success), which will be given by the one that presents the best measure of performance with the least number of layers and neurons.

In order to determine the optimal configuration, it was decided to analyze three network models, each with one, two and three hidden layers. In addition, different numbers of neurons were configured in each layer in each model.

It was found that with four hidden layers the performance measures deteriorated significantly and the MLP models suffered early stops due to overfitting.

The different configurations analyzed are presented in Table IV.

TABLE IV. MLP Configurations Analyzed

	Number of neurons per layer		
Hidden layers	Layer 1	Layer 2	Layer 3
1	From 10 to 150 (step 10)	-	-
2	From 10 to 50 (step 5)	5 fewer neurons than in layer 1	-
3	From 10 to 50 (step 5)	5 fewer neurons than in layer 1	5 fewer neurons than in layer 2

B. Support Vector Machine (SVM) Model

Support Vector Machines (SVM) are part of the supervised learning techniques and are used for both classification and regression. SVMs belong to the family of linear classifications and they find a linear separator or hyperplane [16].

In the classification SVM, the optimal separator hyperplane is defined as the maximum margin separating hyperplane that maximizes its distance from the classes (see Fig. 9).

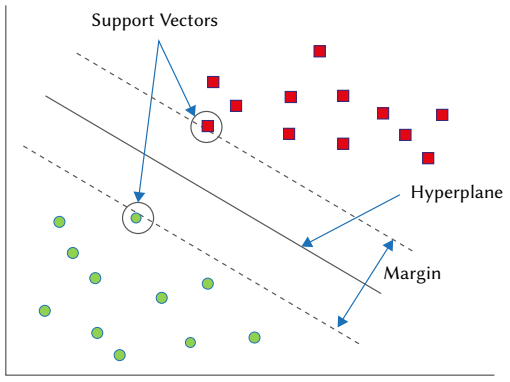

Fig. 9. Maximum margin separating hyperplane for a classification problem with two linearly separable classes.

The concept with which the maximum margin SVM works, is to find the hyperplane separator that is the same distance from the closest examples of each class. Equally, it is the hyperplane that maximizes the minimum distance between the examples of the data set and the hyperplane. Furthermore, it only considers the points that are on the borders of the decision region, these data are the so-called support vectors.

If we have a training dataset $S = \{(x_1, y_1), \dots (x_n, y_n)\}$ where $\boldsymbol{x}_i \in \mathbb{R}^D$ and $y_i \in \{-1, 1\}$, In the case of a linear function, the separating hyperplane will be a linear function:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

Where $\boldsymbol{w} \in \mathbb{R}^D$ is the weight vector orthogonal to the hyperplane and $b \in \mathbb{R}$.

However, the most common formulation of the linear SVM and maximum margin is its primal formulation which is as follows, and corresponds to the quadratic optimization problem whose objective function and the corresponding constrains are:

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|^2$$

s.t.

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \geq 1, i = 1,2, \dots n$$

Separating hyperplanes have two major weaknesses: the requirement for linear separability of the sample and its linear nature. Therefore, in order to extend the SVM concept to non-linear classifiers, a transformation of the input space is performed, using kernel functions $(\Phi(\boldsymbol{x}))$, to another high dimensional feature space in which the data are linearly separable, this procedure is known as the kernel trick [17], [18]. The separation hyperplane then takes the following form:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \Phi(\boldsymbol{x}) + b$$

Likewise, for problems whose training data set is not linearly separable, the Soft-Margin algorithm can be used, which introduces slack variables $(\xi_i \geq 0)$ to relax the condition of the margin, allowing poorly classified observations and making the model more robust [16].

We thus obtain the following optimization problem with objective function and constrains:

$$\min_{\boldsymbol{w},\xi} \left(\frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i \right)$$

s.t.

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0, i = 1,2, \dots n$$

Where the regularization parameter (C) indicates the importance or cost of misclassified instances.

SVM can be applied to regression problems by introducing an alternative loss function. In this work the loss function $\varepsilon - insentitive$ proposed by Vapnik has been used [19], [20], which ignores errors at a certain distance from the real value. In this regression SVM algorithm, some slack variables are also introduced (ξ, ξ^*) which measure the cost of errors in training points, the values of these variables being zero for all points within the band $\pm\varepsilon$ (see Fig. 10).

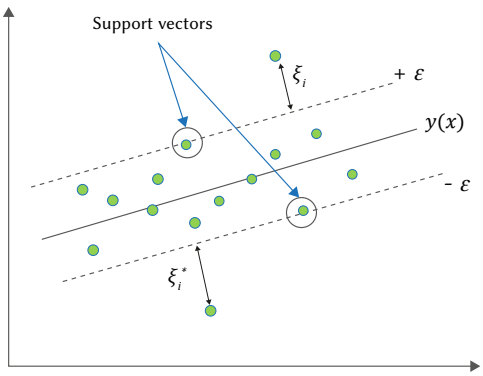

Fig. 10. Regression hyperplane, support vectors, slack variables and $\varepsilon - insentitive$ loss function.

In the case of regression SVM, the quality of the estimate will be given by the loss function (L):

$$L(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{\omega})) = \begin{cases} 0 & if \ |y_i - f(\boldsymbol{x}, \boldsymbol{w})| \leq \varepsilon \\ |y_i - f(\boldsymbol{x}, \boldsymbol{w})| - \varepsilon & otherwise \end{cases}$$

Thus, the optimal regression SVM hyperplane will be given by the following minimization problem:

$$\min_{\boldsymbol{w},\xi,\xi^*} \left(\frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_i (\xi_i^* + \xi_i) \right)$$

s.t.

$$y_i - f(x_i, \omega) \leq \varepsilon + \xi_i^*$$
$$f(x_i, \omega) - y_i \leq \varepsilon + \xi_i$$
$$\xi_i, \xi_i^* \geq 0, i = 1, \ldots n$$

In the same way, in the case of non-linear training data, we will have to perform a non-linear transformation in the input space, to a high dimensional feature space by means of a kernel function.

In our problem, we have used classification SVM with a Soft-Margin algorithm, for the regression SVM models we have used a $\varepsilon - insentitive$ loss function and for both we have implemented a radial base or Gaussian kernel with gamma parameter ($\gamma > 0$).

Therefore, in each case, we must obtain the optimal values for: kernel scale (γ), boxconstrain (C) and $\varepsilon - insentitive$. For this purpose, the grip method or grid search has been used.

C. Finding the Optimal Model

To determine the optimal model for each of the four targets, the following steps were followed:

Step#1. Dataset normalization.

Step#2. Setting the MPL parameters and SVM hyper parameter ranges.

Step#3. Training, validation and testing of all MLP models contained in Table IV and SVM.

The same training and test data were used for this step, for all MLP and SVM configurations. In addition, a total of 10 runs were made, and a statistical study was subsequently carried out to compare the results. In this step, and to save computing time, from the total number of available instances and for each replica, 56 000 instances were sampled.

Step#4. Collection of performance measurements.

For the classification models (solution and probability of success): accuracy (matrix confusion).

For regression models (miss distance and time on cell): mean square error (mse) and R^2.

Step#5. Graphic and statistical analysis of performance measures.

Step#6. Identification of the optimal model for each target.

Step#7. Train the optimal models with a greater number of instances to improve performance measures. A total of 500,000 instances of total available dataset were used.

Step#8. Build the model set to obtain the predictions of the launch solutions.

By applying the Steps# 1-5, the following results have been obtained for each of the four target data:

a) Target: Solution (classification model). In this case and after analyzing the confusion matrices of the different models, the means of the accuracy values obtained are presented in Fig. 11. It can be observed that for the case of MLP with 2 and 3 hidden layers, precision values higher than 0.93 have been obtained.

Furthermore, with 95% confidence, no statistically significant difference was found between MLP with 2 and 3 hidden layers. Therefore, a 2-hidden-layer MLP with 40 and 35 neurons in each layer was taken as the optimal model for this target.

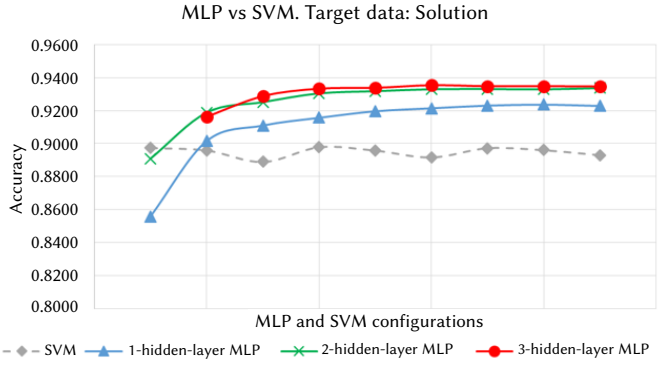

Fig. 11. Solution. Comparative of the accuracy means for the different models.

b) Target: Miss distance (regression model). In this case the value of mean square error and R^2 were used as measures of performance. In Fig. 12, it can be seen that mse minimum value has been reached for a 3-hidden-layer MLP. Furthermore, it was found that statistically and with 95% confidence, the mse values for 3-hidden-layer MLP are statistically lower than the 2-hidden-layer models. Therefore, we will take the 3-hidden-layer MLP with 35-30-25 neurons per layer, as the optimal model for the target data miss distance.

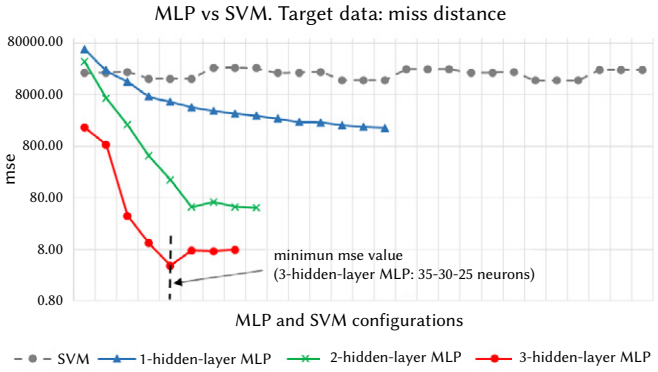

Fig. 12. Miss distance. Comparative of mse means for the different models.

c) Target: % Time in cell (regression model). In this case the value of mean square error and R^2 also were used as measures of performance.

After training the models very high values of mse (higher than 0.001) and poor values of R^2 (lower than 0.80) were obtained.

To improve the results the miss distance was introduced as an input and all models were retrained. With this, the results improved significantly, decreasing the mse by approximately 40% and now obtaining values of R^2 higher than 0.88.

On the other hand, once the graph in Fig. 13 has been created, it can be seen that the best mse results were obtained for 3-hidden layer MLP, specifically for a configuration of 35-30-25 neurons per layer.

d) Target: Probability of success >0.8 (classification model). In this case the same problem was found as for the previous target, the calculated accuracy values in the confusion matrices for the different configurations were lower than 80%, so they cannot be considered as valid for our problem.

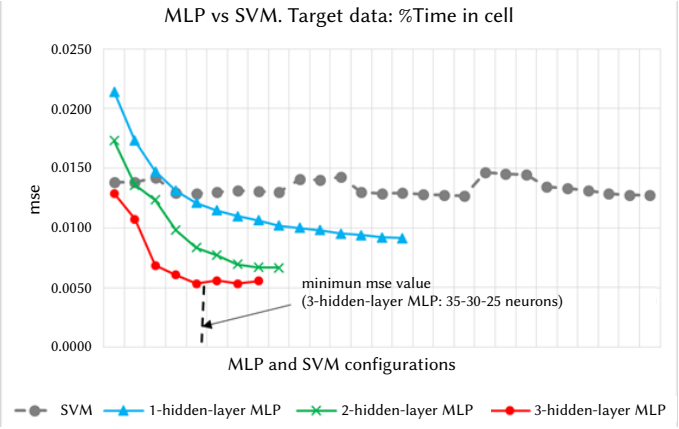

Fig. 13. % Time in cell. Comparative of mse means for the different models.

To improve the results it was introduced as miss distance as a new model input, improving the results reaching precision values close to 90%.

As can be seen in the graph in Fig. 14, all models achieve similar accuracy values, proving that statistically and with 95% confidence, there are no differences between them. Therefore, a 1-hidden-layer MLP of 60 neurons was taken as the optimal model.

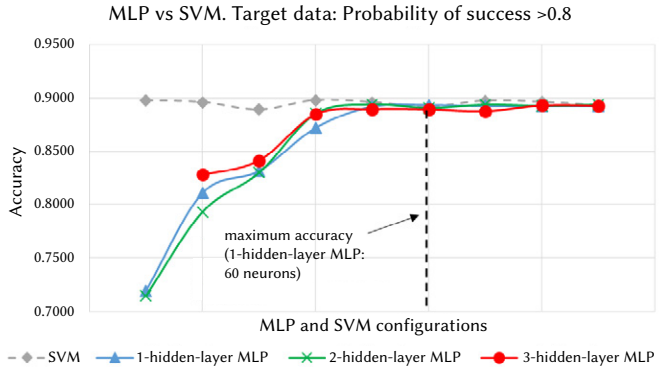

Fig. 14. Probability of success >0.8. Comparative of the accuracy means for the different models.

D. Training, Building the Models Set and Test

Once the optimal models had been found, the following steps were taken as shown in Fig. 15: the optimal models were trained, the final set of models was built and the performance measures of this set were calculated.

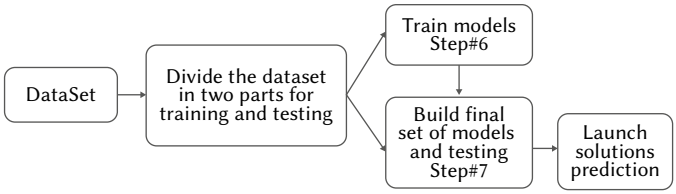

Fig. 15. Training, building the models set and test.

As mentioned in the description of step 3, to save computer time to find the optimal models, small samples of the dataset were used. Therefore, the next step was to train the optimal models found in the previous step with a greater number of instances to improve the performance measurements. For this purpose, we reserved a total of 500,000 instances for models training and the remaining part of the dataset for test the final set of models.

As mentioned above, the time in cell model and the probability of success model were trained by adding the miss target as an input. However, this data is not available in a real scenario, so it was necessary to build a set of MLPs linked between them, so that the output of the model corresponding to miss distance will be part of the input of the time in cell and probability of success models (see Fig. 16).

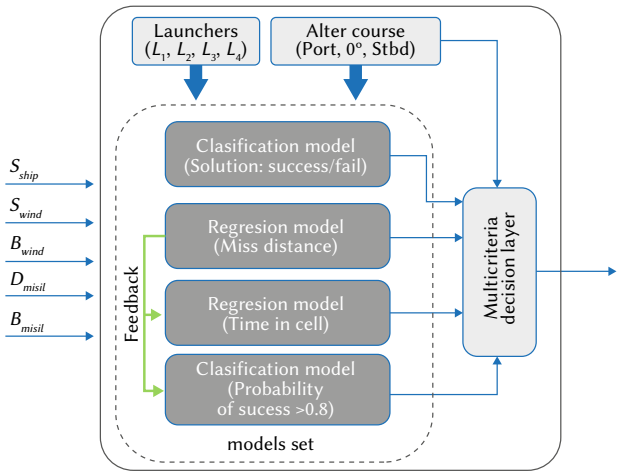

Fig. 16. Final solution. MLP models set.

As there is feedback from the output of the miss distance model to the inputs of others, there is inevitably a propagation of errors, so it is necessary to determine what the values of the performance measures are once the complete set of models have been built. For this purpose, the dataset sample reserved for the MLP set test was used, the test values are contained in Table V.

TABLE V. MLP Models Set Final Performance Measures

Target data	Performance measure
Solution	Accuracy: 96.70%
Miss distance	$mse = 1.42\ yards^2$; $R^2 > 0.99$
% Time in cell	$mse = 0.0043$; $R^2 = 0.91$
Probability of success >0.8	Accuracy: 93.82%

V. Conclusions and Future Work

The set of machine learning models based on neural networks developed in this study can be implemented on any ship and improve the decoy launching solution for any scenario and offer the operator a more evaluated launch response. This only requires the training of the models with the specific data of each ship. In other words, the decoy launch simulator can be substituted by a set of trained MLP and provide real time response to the launch problem.

By having in real time the values of miss distance, % time in cell, probability of success and the need or not for a change of course , they can be used as criteria in a multi-criteria decision algorithm (e.g. Analytic Hierarchy Process -AHP-) to obtain the best possible solution. In other words, following the set of MLP models, a solution manager could be implemented that would automatically evaluate all possible solutions for a scenario without operator intervention.

On the other hand, the model developed is scalable, it is possible to implement on board as many sets of models as there are threat missiles in the operations area and thus have a solution for each of them immediately.

It should also be noted that the work has focused on defense against radio frequency guided missiles, although the methodology proposed here may be applicable to infrared or dual guided threat missiles.

VI. Discussion

The implementation of machine learning models, to solve the problem of decoy launching, opens the way for the introduction of different artificial intelligence techniques in other ship systems, even in those situations where it is necessary to make tactical decisions.

Combined use of naval simulation and AI/machine learning techniques allows us to build models that improve and automatize the decision process on board, which can result in an improvement in the ship survivability and be one of the levers that help to achieve superiority in combat. On the other hand, it is also feasible to automate on-board operational processes, which would make it possible to reduce the required number of crew members and save cost.

References

[1] R. Lord, "Advances in Anti Ship Missile Protection - Naval Countermeasures," Chemring Naval Countermeasures. Salisbure, England, 2006.

[2] W. Sun, "Maneuvering Calculation of Ship Centroid Jamming," in *9th International Conference on Information and Social Science*, 2019, pp. 386–390.

[3] L. F. Galle, "Royal Netherlands Navy The Survivable Frigate," *1o Eur. Surviv. Work.*, 2002.

[4] N. Manju, B. S. Harish, and N. Nagadarshan, "Multilayer Feedforward Neural Network for Internet Traffic Classification," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 6, no. 1, p. 117, 2020.

[5] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[6] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *In IRE WESCON Convention Record, Volume 4, pp. 96–104.* 1960.

[7] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington DC: Spartan Books, 1962.

[8] M. Award and R. Khanna, *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*. Apress Media, LLC, 2015.

[9] D. J. C. Mackay, "A Practical Bayesian Framework for Backprop Networks," *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.

[10] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to bayesian learning," *Proc. Int. Conf. Neural Networks (ICNN'97), Houston, TX, USA*, vol. 3, pp. 1930–1935, 1997.

[11] A. Suliman and B. Omarov, "Applying Bayesian Regularization for Acceleration of Levenberg Marquardt based Neural Network Training," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 1, p. 68-72, 2018.

[12] K. K. Aggarwal, Y. Singh, P. Chandra, and M. Puri, "Bayesian Regularization in a Neural Network Model to Estimate Lines of Code Using Function Points," *Journal of Computer Science*, vol. 1, no. 4, pp. 505–509, 2005.

[13] R.Reed and R. J. Marksll, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Cambridge, Massachusetts, 1999.

[14] J. Heaton, *Introduction to Neural Networks for Java*, 2nd Editio. Heaton Research, Inc., 2008.

[15] D. Stathakis, "How many hidden layers and nodes?," *International Journal of Remote Sensing*, vol. 30, no. 8, pp. 2133–2147, 2009.

[16] C. Cortes and V. Vapnik, "Support-Vector Networks," Kluwer Academic Publishers, 1995.

[17] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[18] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on computatinal learning theory*, 1992, pp. 144–152.

[19] H. Drucker, C. J. C. Surges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," *Advances in Neural Information Processing Systems*, no. June 2013, pp. 155–161, 1997.

[20] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, vol. 17, no. 1, pp. 113–126, 2004.

Ramón Touza Gil

Cmdr. R. Touza is an Officer in the Spanish Navy and a professor at the Spanish Naval Academy. He has received his Master degree in Advanced Data Analysis and Model Building from Complutense University of Madrid and Master in Decision Engineering from Juan Carlos University of Madrid. He is currently a PhD student at Polytechnic University of Cartagena (Spain).

Javier Martínez Torres

Dr. Javier Martínez is a Mathematician and Engineering PhD from the University of Vigo. He is currently an Assistant Professor at the University of Vigo and has participated in more than 20 research projects as principal investigator. He has published more than 50 papers in JCR indexed journals and participate in more than 25 international conferences.

María Álvarez Hernández

She is an associate professor in the Defense University Center – ENM (attached center of the University of Vigo), with a degree in Mathematics from University of Salamanca and PhD. in Statistics and Operations Research for the University of Granada. She has been part of several national projects in Spain, related to the development and research in the statistical field and belongs to the National Network of Biostatistics. The results of her research have been disseminated through a score of papers through indexed scientific journals (within the area of statistics and applied scientific journals) and presentation and participation in dozens national and international scientific conferences.

Javier Roca Pardiñas

Dr. Javier Roca is Associated Professor of the Department of Statistics and Operational Research of University of Vigo. He is specialized in nonparametric regression, and bootstrap inferences techniques with special emphasis on generalized additive models (GAM). Moreover, he is an expert in computational statistics, with important contributions to semiparametric prediction models. His research work generated more than forty publications in international journals of impact in different areas of knowledge such as statistics, computer science, environment, biomedicine, and engineering amongst others. He has also successfully performed as a member of four research projects funded by Spanish Ministry of Science and Innovation devoted to the development of basic and applied mathematical knowledge. He has been principal investigator on several projects with numerous companies and institutions.