DOI: 10.48011/asba.v2i1.1105

A Proposed Methodology for Online Implementation of a Support Vector Machine With Applications in Power Transformers Event Classification

H.J.D. Costa [*], **B.L. Souza** [*], **L.D. Simões** [*], **M.N.O. Aires** [*], **R.P. Medeiros** [**], **O.V. Santana Jr.** [*], **F.B. Costa** [*],

[*] *Federal University of Rio Grande do Norte, Natal, RN, Brazil*
[**] *Federal Rural University of the Semi-Arid, Caraúbas, RN, Brazil*

Abstract: The constant evolution of resources in computational processing and machine learning algorithms, combined with the increasing complexity of embedded systems, made the hardware implementation of machine learning models more viable. This paper proposes a methodology for online implementation of a support vector machine classifier through the development of a simple, concise, and easily adapted algorithm for data classification. The system was validated through the development of an application that classifies disturbances in a power transformer, followed by a comparison with the results obtained with the Library for Support Vector Machines (LIBSVM). Besides the very similar results when compared with the LIBSVM, the proposed methodology achieved high overall accuracy and fast classification time.

Resumo: A constante evolução dos recursos em processamento computacional e dos algoritmos de aprendizagem de máquina, combinado com o aumento da complexidade dos sistemas embarcados, tornou a implementação em *hardware* de modelos de aprendizagem de máquina mais viáveis. Este artigo propõe uma metodologia para implementação *online* de uma máquina de vetor de suporte através do desenvolvimento de um algoritmo simples, conciso e facilmente adaptado para classificação de dados. O sistema foi validado através do desenvolvimento de uma aplicação que classifica distúrbios em um transformador de potência, e em seguida foi feita uma comparação com os resultados obtidos com a *Library for Support Vector Machines* (LIBSVM). Além de resultados muito similares quando comparado com a LIBSVM, a metodologia proposta alcançou uma alta acurácia e um tempo rápido de classificação.

Keywords: machine learning; real-time AI; electrical power systems; support vector machine; methodology.

Palavras-chaves: aprendizagem de máquina; IA de tempo real; sistemas elétricos de potência; máquina de vetor de suporte; metodologia.

1. INTRODUCTION

Machine learning (ML) is a key technology in the 21st century and the main contributing factor for many recent performance boosts in computer vision, natural language processing, speech recognition and signal processing (Roth et al., 2020). It has evolved from the study of computational learning theory and pattern recognition, being the most effective method used in the field of data analytics in order to predict something by devising some models and algorithms (Angra and Ahuja, 2017). ML models allow researchers, engineers, data scientists and analysts to produce reliable and valid results and decisions by the discover of some hidden patterns or features through historical learning's and trends in data. Among the several ML-based algorithms, the support vector machine (SVM) has presented promising results in pattern recognition problems (Karamizadeh et al., 2014).

The SVM was initially introduced by Boser, Guyon and Vapnik in 1992 (Guyon et al., 1992), experiencing a rapid development applied to handwriting recognition and text categorisation, through the Structural Risk Minimization (SRM) method. Currently, the SVM has been applied in several areas, such as robotics, computer vision, pattern recognition, computer security, and biomedics (Yunqian Ma, 2014). For instance, Shah and Bhalja (2013) presents a SVM-based differential protection scheme is proposed which effectively differentiates internal faults with other type of disturbances in a power transformer. Also, Bhargav Yashvantrai Vyas (2015) presents a pattern recognition based fault type identification approach for transmission line protection, which in comparison to a scheme based on artificial neural networks (ANN), the SVM has proved to be a better classifier for protection system.

The advancement in computational processing combined with the increased complexity of embedded systems

⋆ This work was supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) and German Research Foundation (DFG) - PIPC 8881.473092/2019-1 (DFG Grant Number AU 185/72).

has made the implementation of ML models in hardware language more viable in current applications. Nowadays, ML algorithms are often implemented with custom integrated circuits, devices, and systems to achieve high performance. Nevertheless, the development of embedded systems present a variety of challenges as it has to deal with multiple viewpoints, types of requirements, programming languages and forms of description. It is indispensable for the system to be concise, since the diversity of specifications is a major source of inefficiency and failure (Sebastian Bab, 2006).

In recent years, the hardware implementation of ML-based algorithms has been reported in some works in the literature. For instance, a highly-compacted embedded system from the System inPackage type, implemented online on Polymer Electrolyte Membrane Fuel Cell (PEMFC), is presented in (Zhongliang Li, 2019). A SVM algorithm is implemented on a digital signal processor (DSP) for signal prediction in classication and regression applications in (Zabalza et al., 2012), and despite the high consumption of time in some stages, the SVM implementation has been carried out successfully. Furthermore, the implementation of SVMs in DSPs for image tracking was also seen in (Assia Arsalanea, 2018) and (Xinghong Li, 2017), and both of them show that SVM has higher tracking precision and obtained a high speed of target tracking, which makes it possible to achieve real-time performance.

This paper proposes a methodology for online implementation of a support vector machine classifier to be feasible in an embedded system. The proposed SVM classifier was tuned for providing appropriate classification of disturbances in a power transformer, such as internal faults, external faults, and transformer energizations. The disturbances were simulated in a power transformer modeled in ATP software and the signals used as inputs for the SVM classifier are the differential wavelet coefficient energies, which were computed by means of the real-time boundary stationary wavelet transform (RT-BSWT), as seen in (Medeiros and Costa, 2018a). The results revealed the algorithm is simple, efficient and easily adapted for any programming language, also presenting a fast classification time.

2. SUPPORT VECTOR MACHINE

The SVM is a supervised learning ML technique that is based on the statistical learning theory and might be used for both classification or regression problems (Widodo and Yang, 2007). For data classification, the SVM aims to separate data by means of an optimal hyperplane. For this purpose, many formulations can be taken, and one of them is the C-SVM, which achieves it by solving an optimization problem (Chang and Lin, 2011). The SVM is essentially a binary classification algorithm, however, there are methods that make it possible to solve multiclass problems. The fundamentals of both approaches are covered in the remainder of this section.

2.1 Binary classification

The binary SVM classification consists in the partition of an optimal separation hyperplane into two classes, which allows to obtain the support vectors, that delimit the positive and negative hyperplane sides so that data will be classified as belonging to one of these classes (Faceli et al., 2011). The support vectors are positioned in the closest points to the optimal hyperplane, generating a maximized separation margin, represented by $\rho.$, as depicted in Fig. 1.

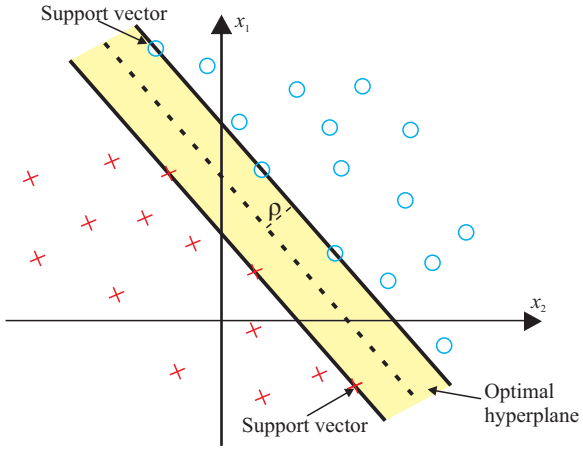

Fig. 1. The optimal separating hyperplane.

Linearly separable data. Let a two class n-feature dataset D with m samples. The samples are assumed to have two classes namely positive class and negative class, with associated labels $y = 1$ and $y = -1$ for positive and negative class, respectively. In the case of linearly separable data, it is possible to determine the hyperplane that separates the given data by

$$\mathbf{w}^T\mathbf{x} + b = 0, \tag{1}$$

where

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2}$$

is the linear classifier. The variables $\mathbf{w}$ and b correspond to a n-dimensional vector and a scalar, respectively, and both are used to define the position of the optimal separating hyperplane (Vapnik, 2013).

Given an unclassified data input $\mathbf{x}$, the decision function uses the sign of the linear classifier to determine the class of $\mathbf{x}$ (C($\mathbf{x}$)), as follows (Vapnik, 2013):

$$C(\mathbf{x}) = \begin{cases} +1, & if\ f(\mathbf{x}) \geq 0, \\ -1, & if\ f(\mathbf{x}) < 0, \end{cases} \tag{3}$$

in which $C(\mathbf{x}) = 1$ and $C(\mathbf{x}) = -1$ correspond to a positive and a negative classes, respectively.

Nonlinearly separable data. SVM can be used in nonlinearly separable data classification applications. In this way, the data to be classified is positioned, by means of kernel functions, in a high-dimension separation hyperplane, allowing a linear classification. In this case, the linear classifier can be rewritten as,

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_s^{(i)} K(\mathbf{x}_s^{(i)}, \mathbf{x}) + b, \tag{4}$$

where l is the number of support vectors, $\mathbf{x}_s^{(i)}$, $y_s^{(i)}$ and α_i are the i-th support vector, its associated label and weight, respectively, and $K(\mathbf{x}_s^{(i)}, \mathbf{x})$ is the kernel function (Vapnik, 2013; Cristianini and Shawe-Taylor, 2000). In the case of linear separable data, the kernel function is the dot product between the unclassified feature vector $\mathbf{x}$ and

the support vector $\mathbf{x}_s^{(i)}$. By replacing it in the classifier equation (4) it is possible to return to the linear classifier equation (2) (Vapnik, 2013).

There are a few types of commonly used kernel functions: the dot product (linear kernel) for linearly separable data, the radial basis function (RBF), and the polynomial and sigmoidal kernel for nonlinearly separable data. Table 1 describes the equations for each of these kernels, being $\mathbf{u}$ and $\mathbf{v}$ two n-feature vectors.

Table 1. Some of the most commonly used kernel functions.

Dot product (linear kernel)	$K(\mathbf{u},\mathbf{v}) = \mathbf{u}\cdot\mathbf{v}$
RBF	$K(\mathbf{u},\mathbf{v}) = e^{-\gamma\|\mathbf{u}-\mathbf{v}\|^2}$
Polynomial	$K(\mathbf{u},\mathbf{v}) = [\tau_1(\mathbf{u}\cdot\mathbf{v}) + \tau_0]^d$
Sigmoidal	$K(\mathbf{u},\mathbf{v}) = tanh[\beta_1(\mathbf{u}\cdot\mathbf{v}) + \beta_0]$

According to Table 1, the $\|\mathbf{u}-\mathbf{v}\|^2$ is the squared Euclidean distance between the two feature vectors and γ, τ_0, τ_1, d, β_0 and β_1 are free parameters (Wang, 2005; Vapnik, 2013).

2.2 Multiclass classification

Two of the main approaches for multiclass classification problems are the one-versus-all and one-versus-one. Given a K-class dataset, these two approaches are described below.

One-versus-all. this method takes one class as positive and rest all as negative and trains the classifier. For instance, if the data has n-classes, it trains n classifiers. Each learner (model) is trained using all patterns of its respective class as positive (+1) and the remaining classes patterns as negative (-1) (Rocha and Goldenstein, 2014). For prediction, the input observations is classified according to the class with the highest response.

One-versus-one. In the training stage this approach will divide the training set in C_2^K subsets and fit C_2^K binary classification models. In the prediction stage, given an unclassified data $\mathbf{u}$, the C_2^K binary classification models will assign a class for $\mathbf{u}$, and the class that receives the most votes will be the class chosen for $\mathbf{u}$ (Rocha and Goldenstein, 2014).

The following example explains how the one-vs-one approach works. In the training stage of a dataset with the classes A, B and C (3 classes), the corresponding training set will be divided into 3 (C_2^3) subsets, with the training examples which contain only the classes A and B, A and C and B and C, respectively. These subsets will train binary classification models for classes A and B, A and C, and B and C. In the prediction stage, given an unclassified data $\mathbf{u}$, the three binary classification models perform the following assignments:

- AB classifier assigns the class A,
- AC classifier assigns the class A,
- BC classifier assigns the class C.

In this case, class A had more votes, thus, the observation $\mathbf{u}$ shall be assigned for class A.

3. PROPOSED METHODOLOGY

This paper proposes a methodology for online implementation of an SVM-based classifier which had been trained offline. For this purpose, considering the wide variety of platforms that enable the training of a classification model, the Library for Support Vector Machines (LIBSVM) (Chang and Lin, 2011) was chosen since it is an open-source, well-established library. The LIBSVM is an integrated open-source ML library for support vector classification, regression, and distribution estimation (one-class SVM), which supports multiple programming languages, such as C/C ++, Python, Java, MATLAB, R, among others. Furthermore, it is also able to perform C-SVM classification, ν-SVM classification, one-class-SVM, ϵ-SVM regression, and ν-SVM regression (Chang and Lin, 2011). In this paper, a C-SVM classification algorithm is carried out, and the LIBSVM was used for purposes of training and validating the model, as well as the proposed method.

Fig. 2 depicts the flowchart with the required stages for online implementation on embedded systems of a SVM classifier, which are: Data acquisition stage (block 1); dataset handling stage (block 2); training stage (block 3); prediction stage (block 4); online validation stage (block 5); and embedded system development, if desired (block 6). Details of each stage are addressed in the remainder of this section.

3.1 Data acquisition stage

The three main methods to get a dataset for training a model for a classification problem are:

- Online open source dataset collections,
- Dataset generated by a computer simulation of a physical system,
- Dataset generated by measurements of a physical system experiment.

In this paper the performance assessment was performed by using a dataset generated by a computer simulation of a physical system.

3.2 Dataset handling stage

Data acquisition bias breaking. Sometimes, data acquisition can carry some biases. For instance, suppose that a model is trained with 1000 feature vectors extracted from measurements and stored in a matrix D, of size $m\times(n+1)$, where m stands for the number of observations, n stands for the number of features, and the $(n+1)$-th column has the labels of each observation. Suppose further that the first half of the measurements corresponds to event A and the other half corresponds to event B. At following, the matrix D must be divided for training and test steps. If, for instance, the first 800 rows of D are assigned to the training set and the rest to the test set, a bias will arise even though the original dataset does not show it. This bias can lead to bad performance. Therefore, it is a good practice to shuffle rows of D at the beginning of the dataset handling stage.

Split dataset into training and test set. In order to validate if the trained model is able to perform well with unseen data, it is needed to split the dataset into two subsets: training set and test set (Suthaharan, 2016). It is good practice to leave 20 to 30% of the dataset for testing purposes.

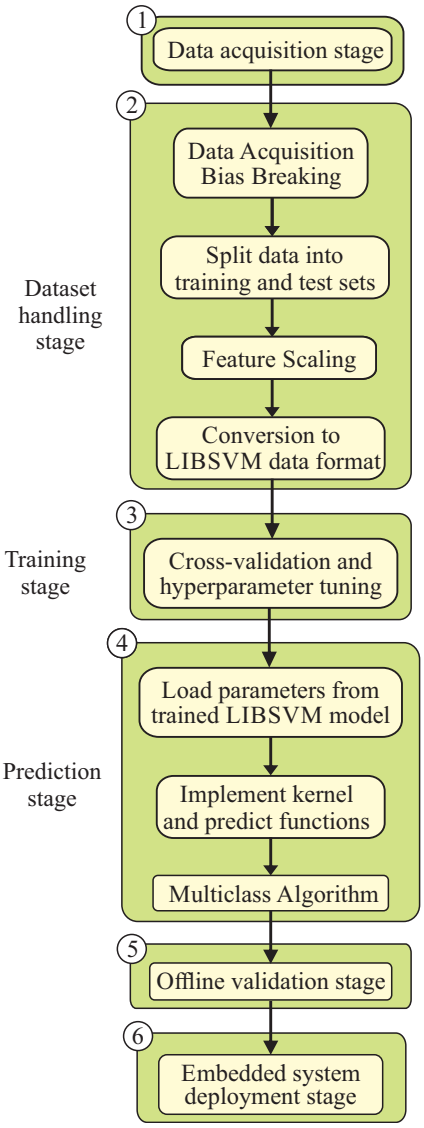

Fig. 2. Proposed methodology methodology for hardware implementation of a support vector machine.

Feature scaling. The feature scaling allows, by means of mathematical handlings, that the features present the same value range (Reddy et al., 2016). This approach speed up the training stage and can lead to better results (Li and Liu, 2011). There are several feature scaling methods in literature, such as decimal scaling, min-max, zero-mean, sigmoidal, softmax and max normalization (Li and Liu, 2011). For instance, taking $x_k^{(i)}$ a non-scaled variable, its respective scaled version can be obtained with the min-max [0,1] and zero-mean normalization methods, respectively, as follows:

$$x_k'^{(i)} = \frac{x_k^{(i)} - min(x_k)}{max(x_k) - min(x_k)}, \tag{5}$$

$$x_k'^{(i)} = \frac{x_k^{(i)} - \overline{x}_k}{\sigma_k}, \tag{6}$$

where $min(x_k)$, $max(x_k)$, $\overline{x}_k$ and σ_k are the minimum, the maximum, the average and the standard deviation value of the feature k in the dataset, respectively (Li and Liu, 2011). The feature scaling step should be carried after the dataset is divided into training and test set, and even more relevant: it should be performed accounting only the feature scaling parameters from the training set, since using both training and test data for normalization will cause data leakage. This causes the model to artificially inflate its performance in the test stage, however, it makes its performance worse when facing new, real world data (Ameisen, 2020).

Original dataset to LIBSVM data format conversion. LIBSVM stores data in a quite particular format. Therefore, before running the training stage, it is needed to convert the original data to LIBSVM format. In our case, for instance, it was needed to make the conversion from comma-separated value (CSV) to the LIBSVM format. More information about this data type conversion can be found at (Chang and Lin, 2019).

3.3 Training stage

Once dataset handling is finished, the training stage is then carried out. The model is trained with the LIBSVM `svmtrain` function, in which is loaded the training set (observations and its respective labels) and is defined: the SVM algorithm and the kernel function and its hyperparameters (Chang and Lin, 2011). Hyperparameters are parameters that are not directly learnt within estimator, i.e., free parameters. For instance, in a C-SVM classification estimator with a RBF kernel, the regularization parameter C and the RBF free parameter γ are the hyperparameters. The parameters C and γ must be strictly positive, and they have a crucial role in learning the dataset behavior. For too large values of C or γ, overfitting is likely to occur, which means that the model learns too well the training set data behavior (high accuracy classification), however it fails in generalization capability (the hability to generalize the model to unseen data), which is a key concept in machine learning models assessment. Conversely, very small values of C or γ can also cause bad performance, thus yielding an approximately linear separation between the data, causing a bad performance for both training and unseen (test) data (Johnson and Yadav, 2017).

In order to achieve better performance, the hyperparameters need to be tuned. The following section describes a method commonly used to adjust hyperparameters, known as grid search.

Tuning the hyper-parameters of an estimator. The grid search is a technique used to optimize the SVM parameters through cross-validation (CV), being able to identify the best combination between the hyperparameters such that the classifier predicts the unknown data and achieves better accuracy, also avoiding overfitting (Iwan Syarif, 2016). In this technique, the CV error is computed with different values of C and γ, and the best CV accuracy (or the lowest CV error) is used to train a final model of the SVM to be used in the test stage.

3.4 Prediction stage

After the training stage, it is obtained a file containing information about the classification parameters, such as the support vectors and the kernel. Therefore, an algorithm for classification of this new information is required, which must be of simple implementation in order to make its adaptation more viable to other programming

languages compatible with embedded systems (Lin et al., 2019). For this purpose, two main functions were created: predict and kernel. However, it was also necessary create a multiclass classification algorithm for handling cases where there are more than two classes, for which the predict function is not sufficient.

Kernel function. The `kernel` function contains the equations that define the chosen kernel, for instance, the kernel equations from Table 1 or any other user-defined kernel equation.

Predict function. The `predict` function receives as inputs the trained model and an unclassified data vector, and implements the routines of the classification hyperplane, the classifier and the decision function, in addition to using the `kernel` function. This predict algorithm is adaptable for any kernel, being able to return the classification results according to the model and entered values.

The employed C-SVM binary prediction algorithm is shown in Algorithm 1. The trained *model* is loaded with the following parameters: support vectors and their respective labels and weights, γ, positive and negative class labels. Based on these parameters and on the hyperplane's equations, the algorithm computes a classification score, and according to the sign of the obtained value, the function decides whether the input vector, or observation, is on the positive or negative side of the hyperplane.

One-vs-one function. The methodology discussed in Section 2.2 for one-vs-one multiclass classification is directly applied to the algorithm, whereas the same data is received as input in each binary classification model, and the final output (class) of this function is the one that has the most votes among the results obtained with the aforementioned `predict` function.

3.5 Offline validation stage

This validation stage was performed comparing the classification scores and the confusion matrices from both our implemented `predict` function and the LIBSVM `svmpredict` function. The purpose was to obtain results approximately equal between both algorithms. Details of this comparison are addressed in the next section.

3.6 Embedded system deployment

After the algorithm validation, the same is available to be adapted to the programming language of the respective embedded system in which is intended to be used. This task was not tested in this paper. However, considering the procedure used in the previous steps, it is expected the embedding process of the SVM algorithm to be simple and feasible. Nevertheless, MATLAB platform was used as an alternative for SVM algorithm development, by allowing the implementation of the preprocessing steps, as well as the `kernel` and the `predict` functions for the performance assessment.

4. PERFORMANCE ASSESSMENT

In this paper, the proposed SVM algorithm was used with purpose of classify electrical disturbances in a power transformer. In this way, an electrical power system was simulated in Alternative Transient Program (ATP), as depicted in Fig. 3. The system consists of a power

Algorithm 1 C-SVM binary prediction algorithm

procedure PREDICT($model$, $\mathbf{u}$)
  ▷ Classification score calculation
  $score = 0$
  for $i = 1, k$ **do** ▷ k is the number of support vec.
    $score = score + \alpha_i y^{(i)} * \texttt{kernel}(\mathbf{u}, \mathbf{x(i)}, \gamma)$
  end for
  $score = score + b$
  ▷ Decision
  if $score \geq 0$ **then**
    $out =$ positive class
  else
    $out =$ negative class
  end if
end procedure
procedure KERNEL($\mathbf{x}_1$, $\mathbf{x}_2$, γ)
  $sim = 0$
  $norm = 0$
  $difference = 0$
  $square = 0$
  $add = 0$
  for $i = 1, n$ **do** ▷ n is the number of features
    $difference = \mathbf{x}_1(i) - \mathbf{x}_2(i)$
    $square = difference^2$
    $add = add + square$
  end for
  $norm = sqrt(add)$
  $sim = exp(-\gamma * norm^2)$
  $out = sim$
end procedure
procedure MAIN()
  repeat
    load $\mathbf{u}$
    predict($model, \mathbf{u}$)
  until *stop*
end procedure

transformer with their primary and secondary windings connected to the 230 kV and 69 kV busbars, respectively. The currents measured in the current transformers CT1 and CT2 are inputs for the relay, where they are sampled at a sampling frequency of 15360 Hz (256 samples per cycle of 60 Hz). Details about the preprocessing steps, as well as the system parameters, are adressed in (Medeiros and Costa, 2018a).

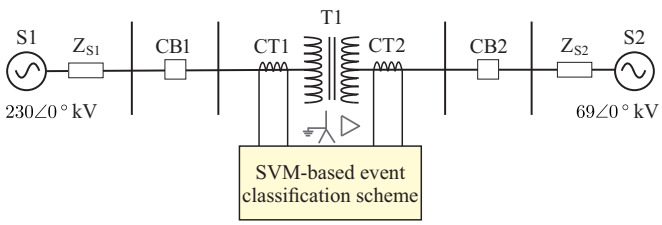

Fig. 3. Electrical system single phase diagram employed to validate the proposed methodology.

The following databases were used to evaluate the performance of the method:

- Database 1 (external faults): AG, AB, AC, ABG, ACG and ABC faults, on both high and low voltage sides of the power transformer, with fault inception

angle $\theta_f = \{0, 30, 60, 90, 120, 150, 180\}$ electrical degrees and fault resistance $R_f = \{1, 2, 3, ..., 9, 10\}$ Ω (840 records).

- Database 2 (internal faults): AG, BG, CG, AB, BC, AC, ABG, BCG, ACG and ABC faults, on the high and the low voltage windings of the power transformer, with $\theta_f = \{0, 30, 60, 90, 120, 150, 180\}$ electrical degrees and $R_f = \{10, 20, 30, ..., 90, 100\}\Omega$ (1400 records).
- Database 3 (critical internal faults): turn-to-turn faults on the phase A wye winding; turn-to-turn faults on the delta winding between phases A and B; turn-to-earth faults on the phase A wye winding; and turn-to-earth faults on the delta winding between the A-to-B winding and the earth. The percentage of turns affected by the fault varies with $e = \{1, 2, 3, ..., 98\}\%$ (392 records)
- Database 4 (transformer energizations): switching performed on the high voltage side (230 kV), with the secondary terminal opened, and varying the high voltage circuit breaker closing time at angles $\theta_s = \{0, 1, 2, ..., 179, 180\}$ electrical degrees, considering the presence and the absence of residual flux for each assessed angle (362 records).

The proposed method is feeded with the operating and restraining wavelet coefficient energies of the currents, which were proposed in (Medeiros and Costa, 2018b). When any disturbance (internal fault, external fault or transformer energization) is detected, the differential energies sudden increase and the SVM-based classification algorithm is enabled. At following, a vector containing the first eight post-disturbance samples of the operating and restraining wavelet energies of the A, B, and C phases, and also of the negative sequence, is stored, which yields 64 input features for each observation. These features are used as inputs for the predict function so that already trained SVM indicates the type of fault that occurred.

The kernel chosen for the training of SVM in LIBSVM was the RBF, thus, being it necessary to assign values to C and γ parameters. A grid search with 10-fold cross-validation was then performed in order to obtain the combination of values for C and γ that achieve the highest accuracy during the cross-validation, as described in Section 3.3. For the binary model, $C = 1$ and $\gamma = 100$ were obtained, and for the multiclass model, $C = 10$ and $\gamma = 10$ in two binary classifiers (internal faults vs. transformer energizations dataset and external faults vs. transformer energizations dataset) and $C = 1000$ and $\gamma = 100$ in the third one (external faults vs. internal faults dataset).

4.1 Binary algorithm performance evaluation

With C and γ parameters defined and the data generated, the classifier performance is evaluated. Fig. 4 depicts the confusion matrix obtained for the binary classifier, with distinction between internal fault and transformer energization cases. The main diagonal of the confusion matrix represents the percentage of cases correctly classified for each detected event, whereas the other elements are associated to the misclassified cases.

According to Fig. 4, the proposed SVM classification model obtained a 100% success rate in discriminating between internal fault and transformer energizations and

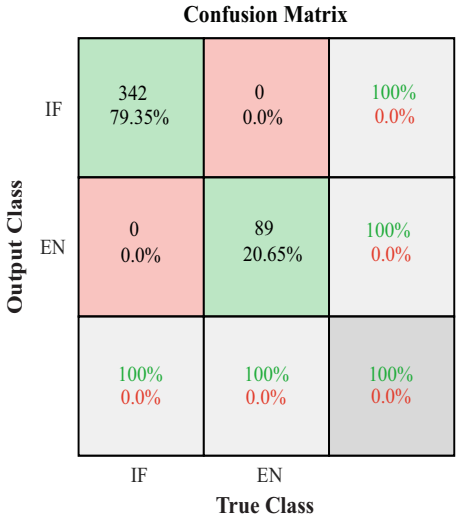

Fig. 4. Proposed SVM algorithm performance distinguishing between internal faults (IF) and transformer energizations (EN).

no false positive or negative were find. These events were correctly classified in a maximum time delay of 8 samples after the event detection, which makes the proposed method quite fast for electrical disturbance classification purposes.

The performance of the proposed function was validated against the LIBSVM `svmpredict` function, which has its results for the same test set depicted in 5. Likewise, to further investigate the performance of the implemented `predict` function, a comparison with the classification scores obtained from the LIBSVM `svmpredict` function was made. These classification scores are used to define whether an observation at hand belongs to a class or another. Table 2 depicts the classification scores from the LIBSVM and the proposed method for 10 random observations from the test set, as well as its predicted classes (diagnosis). For all the shown cases, the classification scores are equal, therefore, the diagnosis is also the same for both the LIBSVM and our `predict` function, hence validating the good performance of the proposed methodology when compared to the LIBSVM.

Table 2. Classification score comparison.

LIBSVM	Proposed Method	Diagnosis
0,99784	0,99784	EN
1,00053	1,00053	EN
-1,75508	-1,75508	IF
-3,78615	-3,78615	IF
-0,97821	-0,97821	IF
-2,32951	-2,32951	IF
0,99958	0,99958	EN
-0,99555	-0,99555	IF
-1,16307	-1,16307	IF
0,99918	0,99918	EN

4.2 Multiclass algorithm performance evaluation

The choice of C and γ parameters in the multiclass algorithm is different of what is accomplished in the binary case. The method chosen was the one-vs-one and it was necessary to make the grid search for three binary

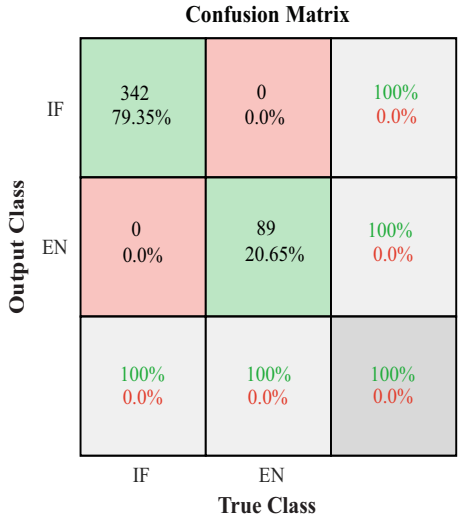

Fig. 5. LIBSVM `svmpredict` function performance distinguishing between internal faults (IF) and transformer energizations (EN).

classifiers that would be the inputs for the predictive one-vs-one function. Coincidentally, the values of C and γ obtained by the three classifiers were very similar and reached a high accuracy.

Fig. 6 illustrates the confusion matrix obtained for the multiclass one-vs-one classifier, with distinctions among the external faults, internal faults, and transformer energizations.

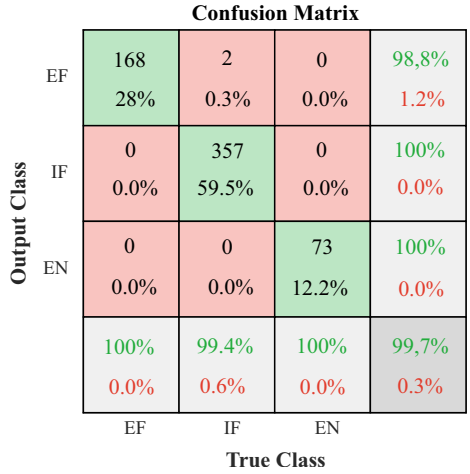

Fig. 6. Proposed SVM algorithm performance distinguishing among external faults (EF), internal faults (IF) and transformer energizations (EN).

According to Fig. 6, the proposed SVM model showed an expressive success rate of 99.7% in discriminating among internal faults, external faults, and transformer energizations. There were only 2 cases misclassified (two internal faults misclassified as external faults). Furthermore, all events were correctly classified with a maximum time delay of 8 samples after their detection.

Similarly to the comparison made to validate the results for the binary classification, another comparison was carried out for the multiclass case. The confusion matrix obtained for the LIBSVM `svmpredict` function is depicted in Fig. 7. Even though the performance of the LIBSVM `svmpredict` was slightly distinct, this could be due to the different internal implementation of the one-vs-one strategy from the LIBSVM, since it receives only one pair of C and γ as input values, differently from our implemented one-vs-one, which carries a diverse pair of C and γ for each internal binary classifier. Nevertheless, it possible to validate the performance of the implemented multiclass algorithm, since the results from both confusion matrices are very similar.

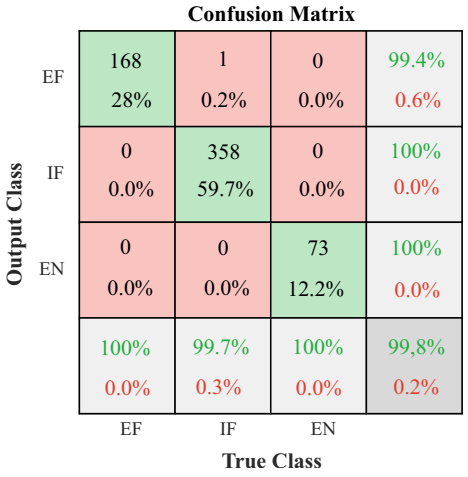

Fig. 7. LIBSVM `svmpredict` function performance distinguishing among external faults (EF), internal faults (IF) and transformer energizations (EN).

5. CONCLUSION

The efficiency and feasibility of the proposed methodology for implementing the online SVM were validated in the application of fault classiffication in power transformers. A point that makes the use of this method very accessible is its association with LIBSVM training, a platform that allows all stages of training to be carried out in a simple and automatic way, in addition to providing support for its use in different programming languages. Despite this work addressing an application for electrical engineering, the algorithms used were tested with different types of databases, varying the number of samples, features, classes, and their efficiency was likewise validated. In the same way that the function was developed and tested in MATLAB, this proposed methodology can be easily adapted for languages like C/C++, Python, among others, and later embedded on hardware devices, allowing it to be employed for many other applications with real-time requirements.

REFERENCES

Ameisen, E. (2020). *Building Machine Learning Powered Applications*. O'Reilly Media, Inc., Sebastopol, CA, USA, 1st edition.

Angra, S. and Ahuja, S. (2017). Machine learning and its applications: A review. In *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, 57–60.

Assia Arsalanea, Noureddine El Barbrib, A.T.A.K.K.R.A.H. (2018). An embedded system based on dsp platform and pca-svm algorithms for

rapid beef meat freshness prediction and identification. *Computers and Electronics in Agriculture*, 152.

Bhargav Yashvantrai Vyas, Rudra Prakash Maheshwari, B.D. (2015). Pattern recognition application of support vector machine for fault classification of thyristor controlled series compensated transmission lines. *Journal of The Institution of Engineers*, 97, 175–183.

Chang, C.C. and Lin, C.J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2, 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Chang, C.C. and Lin, C.J. (2019). *LIBSVM FAQ*. National Taiwan University. Viewed 15 march 2020, `<https://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#f307>`.

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines & Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge.

Faceli, K., Lorena, A.C., Gama, J., and de Carvalho, A.C.P.d.L.F. (2011). *Inteligência artificial: uma abordagem de aprendizado de máquina*. LTC, Rio de Janeiro, 1st edition.

Guyon, I., Vapnik, V., Boser, B., Bottou, L., and Solla, S. (1992). Capacity control in linear classifiers for pattern recognition. In *Conference B*, Proceedings - International Conference on Pattern Recognition, 385–388. Institute of Electrical and Electronics Engineers Inc., United States. doi:10.1109/ICPR.1992.201798. 11th IAPR International Conference on Pattern Recognition, IAPR 1992 ; Conference date: 30-08-1992 Through 03-09-1992.

Iwan Syarif, Adam Prugel-Bennett, G.W. (2016). Svm parameter optimization using grid search and genetic algorithm to improve classification performance. *TELKOMNIKA*, 14, 1502–1509.

Johnson, J.M. and Yadav, A. (2017). Complete protection scheme for fault detection, classification and location estimation in HVDC transmission lines using support vector machines. *IET Science, Measurement and Technology*, 11(3), 279–287. doi:10.1049/iet-smt.2016.0244.

Karamizadeh, S., Abdullah, S.M., Halimi, M., Shayan, J., and j. Rajabi, M. (2014). Advantage and drawback of support vector machine functionality. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, 63–65.

Li, W. and Liu, Z. (2011). A method of SVM with normalization in intrusion detection. *Procedia Environmental Sciences*, 11(PART A), 256–262.

Lin, C.J., Que, Z., and Júnior, P.R.M. (2019). *README*. GitHub. Viewed 15 march 2020, `<https://github.com/cjlin1/libsvm/blob/master/README>`.

Medeiros, R.P. and Costa, F.B. (2018a). A wavelet-based transformer differential protection: Internal fault detection during inrush conditions. *IEEE Transactions on Power Delivery*, 33(6), 2965–2977.

Medeiros, R.P. and Costa, F.B. (2018b). A wavelet-based transformer differential protection with differential current transformer saturation and cross-country fault detection. *IEEE Transactions on Power Delivery*, 33(2), 789–799.

Reddy, R.R., Ramadevi, Y., and Sunitha, K.V.N. (2016). Effective discriminant function for intrusion detection using svm. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1148–1153.

Rocha, A. and Goldenstein, S.K. (2014). Multiclass from Binary: Expanding One-Versus-All, One-Versus-One and ECOC-Based Approaches. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2), 289–302.

Roth, W., Schindler, G., Zöhrer, M., Pfeifenberger, L., Peharz, R., Tschiatschek, S., Fröning, H., Pernkopf, F., and Ghahramani, Z. (2020). Resource-efficient neural networks for embedded systems.

Sebastian Bab, Bernd Mahr, G.H.S.H. (2006). *Embedded Systems – Modeling, Technology, and Applications*. Springer.

Shah, A.M. and Bhalja, B.R. (2013). Discrimination between internal faults and other disturbances in transformer using the support vector machine-based protection scheme. *IEEE Trans. Power Del.*, 28(3), 1508–1515.

Suthaharan, S. (2016). *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Springer US.

Vapnik, V.N. (2013). *The Nature of Statistical Learning Theory*. Springer Science & Business Media.

Wang, L. (ed.) (2005). *Support Vector Machines: Theory and Applications*. Springer-Verlag Berlin Heidelberg, 1st edition.

Widodo, A. and Yang, B.S. (2007). Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6), 2560–2574.

Xinghong Li, Qian Xiang, J.C. (2017). Application of dual dsp target tracking system based on svm. *Computers and Electronics in Agriculture*.

Yunqian Ma, G.G. (2014). *Support Vector Machines Applications*. Springer International Publishing.

Zabalza, J., Ren, J., Clemente, C., Di Caterina, G., and Soraghan, J. (2012). Embedded svm on tms320c6713 for signal prediction in classification and regression applications. In *2012 5th European DSP Education and Research Conference (EDERC)*, 90–94.

Zhongliang Li, Rachid Outbib, S.G.D.H.S.J.A.G.S.R. (2019). Fault diagnosis for fuel cell systems: A data-driven approach using high-precise voltage sensors. *Renewable Energy*, 135.