Comparing Sparsification Techniques in the Design of Kernel Prototype-Based Classifiers

David N. Coelho
Engenharia da Computação
Universidade Federal do Ceará (UFC)
Sobral, CE, Brasil
david.coelho@sobral.ufc.br

Guilherme A. Barreto
Pós-Graduação em Eng. de Teleinformática (PPGETI)
Universidade Federal do Ceará (UFC)
Fortaleza, CE, Brasil
gbarreto@ufc.br

***Abstract*—This article addresses the construction of kernelized prototype-based models through sparsification methods. To this end, it extends the algorithm proposed by [1], which employs the ALD method for sparsification and the nearest neighbor classifier based on the selected prototypes. Three additional sparsification methods, namely coherence, novelty, and surprise, are incorporated and compared to the ALD-based approach, along with the use of the weighted K-nearest neighbors algorithm. The results show improvements in the performance of the original algorithm across various datasets, highlighting the strong influence of the chosen sparsification method and kernel function on the classifier's accuracy rates and the number of selected prototypes.**

***Index Terms*—Prototype-based classifiers, sparsification, kernel methods, K-nearest neighbors.**

I. Introduction

Kernel-based methods have been introduced with the aim of developing nonlinear versions of linear supervised or unsupervised machine learning algorithms. The underlying idea of these methods is to apply a kernel function $\kappa(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ to any pair of vectors $(\mathbf{x}_i, \mathbf{x}_j)$ so that the result can be interpreted as an inner product between two vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$, where $\phi : \mathcal{X} \rightarrow \mathcal{F}$, and $\mathcal{F}$ is a high-dimensional (possibly infinite dimensional) feature space [2]: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$. Since $\phi(\cdot)$ is usually unknown in practical applications, inner products in the feature space are computed through the kernel function without using the feature vectors $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ directly. This property of kernel methods is known as the *kernel trick* [3].

In the field of kernel adaptive filtering (KAF), kernel least mean squares (KLMS) [4] and kernel recursive least squares (KRLS) [5] are some examples of algorithms that benefit from the kernel trick in order to create nonlinear versions of classical algorithms. However, the principal bottleneck of this class of kernelized algorithms is their growing structure with each new sample. So, sparsification methods were developed in order to control a model's structure size by verifying if a given sample is important or redundant to the existing model [6], [7], [5], [8].

The process of *kernelization* has also been applied to prototype-based algorithms, producing their kernelized versions, such as the kernel self-organizing maps (KSOM) [9] and the kernel learning vector quantization (KLVQ) [10]. It is noteworthy, however, that the performance of these models, when used for classification, is highly dependent on the number of labeled prototypes. In scenarios where all the data are available for offline processing, a set of prototypes can be either optimally determined using evolutionary algorithms [11] or be added/removed adaptively [12]. However, in most applications, that number is set by trial and error or exhaustive grid search.

Bearing this issue in mind and based on ideas from the KAF field, a simple design scheme for building kernelized prototype-based classifiers by means of the *approximate linear dependence* (ALD) method was developed in [1]. In this method, the built dictionary is used to classify new samples using a kernelized nearest neighbors scheme. Unlike this previous work, in the current paper, it is proposed an alternative approach that generalizes the ALD-based method by incorporating different sparsification procedures and by employing the *kernel-weighted K-nearest neighbors* (KWKNN) algorithm instead of the standard nearest neighbor method. A set of experiments with benchmarking datasets confirm that using other sparsification criteria and the KWKNN classifier can enhance the model's accuracy without compromising its size.

The remainder of this paper is organized as follows. In Section II, the fundamentals of kernel prototype-based classifiers are reviewed. In Section III, the sparsification methods are discussed and their equations are shown. In Section IV, the proposed approach is summarized. In Section V, the datasets and methods used for the algorithm evaluation are detailed. The simulation results are reported and discussed in Section VI. The paper is concluded in Section VII.

II. Kernel Prototype-based Classifiers

In the context of classification, which is a supervised learning task, a data set is given as a set of tuples (samples) $\{(\mathbf{x}_n, c_n) \in \mathbb{R}^p \times \{1, \ldots, C\}\}_{n=1}^N$, where $\mathbf{x}_n$ is the n-th input feature vector, c_n is its associated class label, and C is the finite and predefined number of classes ($C \ll N$).

Depending on the problem, there are several ways to encode class labels. In this paper, a column vector $\mathbf{y}_n \in \mathbb{R}^{C \times 1}$ is built by means of the one-hot encoding scheme. As such,

This research was funded by CAPES (Finance Code 001) and CNPq (Grant No. 313251/2023-1).

the component of the output vector referring to the class to which the input data belongs has the value $+1$, while the other components have the value -1. In this sense, a data set is given as a collection of input-output pairs $\{(\mathbf{x}_n, \mathbf{y}_n) \in \mathbb{R}^p \times \mathbb{R}^C\}_{n=1}^N$.

In batch learning, the data set of N samples is previously stored and can be divided into two disjoint subsets: the training set (with N_{tr} samples) and the test set (with N_{ts} samples). The classifier model H is built from the training set, and the test set is used to validate it. To analyze the classifiers in terms of the generalization ability to unseen samples, the classification batch error measure is given by

$$E_b = \frac{1}{N_{ts}} \sum_{n=1}^{N_{ts}} \Theta(c_n, \hat{c}_n), \tag{1}$$

where $\hat{c}_n = H(\mathbf{x}_n)$ is the classifier's output prediction, and the loss function $\Theta(.,.)$ is either equal to 0, if $c_n = \hat{c}_n$, or 1, otherwise.

Of great interest to this work, prototype-based algorithms are also called *competitive learning* algorithms within the field of artificial neural networks [13]. Models and algorithms based on the principle of competitive learning include the neural gas (NG) and its variants [14], the family of learning vector quantization (LVQ) algorithms [15] and Kohonen's self-organizing maps (SOM) [16].

Competitive learning algorithms learn from data a mapping from a continuous input space $\mathcal{X}$ onto a discrete set $\mathcal{A}$ of Q prototypes. The map $q^*(\cdot) : \mathcal{X} \to \mathcal{A}$, defined by the set of weight vectors $\mathbf{W} \in \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_Q\}, \mathbf{w}_q \in \mathbb{R}^p \subset \mathcal{X}$, assigns to each input vector $\mathbf{x}(t) \in \mathbb{R}^p \subset \mathcal{X}$ a winning prototype $q^*(t) \in \mathcal{A}$, determined by a measure of (dis)similarity such as

$$q^*(t) = \arg\min_{\forall q} \|\mathbf{x}(t) - \mathbf{w}_q(t)\|_2^2, \tag{2}$$

where $\|\cdot\|_2$ denotes the Euclidean distance and t symbolizes a discrete time step associated with the iterations of the algorithm. In order to develop nonlinear versions of these prototype-based models, one can use the kernel trick. In this regard, the search for the winning prototype, as originally shown in Eq. (2), becomes

$$\begin{aligned} q^*(t) &= \arg\min_{\forall q} \|\phi(\mathbf{x}(t)) - \phi(\mathbf{w}_q(t))\|_2^2, \\ &= \arg\min_{\forall q} J_q(\mathbf{x}(t)), \end{aligned} \tag{3}$$

where $J_q(\mathbf{x}(t))$ is a cost function. Using some linear algebra and the kernel trick, this expression can be developed as

$$\begin{aligned} J_q(\mathbf{x}(t)) &= \|\phi(\mathbf{x}(t)) - \phi(\mathbf{w}_q(t))\|_2^2 \\ &= (\phi(\mathbf{x}(t)) - \phi(\mathbf{w}_q(t)))^T (\phi(\mathbf{x}(t)) - \phi(\mathbf{w}_q(t))) \\ &= \phi(\mathbf{x}(t))^T \phi(\mathbf{x}(t)) + \phi(\mathbf{w}_q(t))^T \phi(\mathbf{w}_q(t)) \\ &\quad - 2\phi(\mathbf{x}(t))^T \phi(\mathbf{w}_q(t)) \\ &= k(\mathbf{x}(t), \mathbf{x}(t)) + k(\mathbf{w}_q(t), \mathbf{w}_q(t)) \\ &\quad - 2k(\mathbf{x}(t), \mathbf{w}_q(t)). \end{aligned} \tag{4}$$

Different kernel functions will lead to different kernelized selection of winning prototypes in Eq. (4). The kernel functions used in this paper are shown in Table I. The Gaussian, Cauchy and Log kernel functions have hyperparameters whose values affect directly the quality of the implemented nonlinear mapping. Thus, fine-tuning of these hyperparameters is fairly important to the success of the classification task.

TABLE I
KERNEL FUNCTIONS USED IN THIS PAPER AND THEIR HYPERPARAMETERS

Kernel	Expression	Hyperparameters
Linear	$\kappa(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v}$	-
RBF	$\kappa(\mathbf{u}, \mathbf{v}) = \exp\left(-\frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{2\gamma^2}\right)$	γ (scale parameter)
Cauchy	$\kappa(\mathbf{u}, \mathbf{v}) = \left(1 + \frac{\|\mathbf{u}-\mathbf{v}\|_2^2}{\gamma^2}\right)^{-1}$	γ (scale parameter)
Log	$\kappa(\mathbf{u}, \mathbf{v}) - \log\left(1 + \frac{\|\mathbf{u}-\mathbf{v}\|_2^l}{\gamma^2}\right)$	γ and l

After the training part of a prototype-based model (either by selecting relevant prototypes from a dataset or updating a pre-defined number of prototypes), the K-nearest neighbors methods can be used for classification. These methods represent one of the simplest and most intuitive nonparametric techniques in the field of statistical classification.

In its simplest version, considering just one neighbor, a new observation inherits the same label as the closest sample from a dictionary composed either of training samples or prototypes. This method is also known as the nearest neighbor classifier (NNC) or nearest prototype classifier (NPC) [13]. A first extension of this idea is to use more than one neighbor, resulting in the K-nearest neighbors (KNN) variant. In this case, the K closest observations within the learning set are used for the sake of classification [17]. The decision is in favor of the most frequent class label among the ones in the current set of K closest observations. The drawback of this extension is that K is a hyperparameter that must be optimally chosen before training, and different values of K for the same sample can lead to different classification outputs.

More sophisticated methods can be derived from this majority voting KNN strategy. The weighted KNN (WKNN) [18], [17] is based on the idea that prototypes which are closer to a new observation should get a higher weight in the class prediction of this observation.

The first step of this method is to calculate the distances $d(\cdot,\cdot) \in \mathbb{R}$ of the new sample to the model's prototypes. Then, one must select the $K+1$ nearest neighbors and their distances from this sample. After that, these distances have to be transformed, according to an arbitrary function, into similarity measures which can be used as weights. This arbitrary function must hold the following properties:

- $f(d(\cdot,\cdot)) \geq 0$ for all $d(\cdot,\cdot)$
- $f(d(\cdot,\cdot))$ gets its maximum for $d(\cdot,\cdot) = 0$
- $f(d(\cdot,\cdot))$ decreases monotonically for $d(\cdot,\cdot) \to \infty$

The rectangular function (where all distances have the same weight, turning the KNN into a special case of WKNN) and the triangular function, which can be defined as

$$f(d(\cdot,\cdot)) = 1 - |d(\cdot,\cdot)|, \tag{5}$$

are some examples of functions that can be used for the transformation.[1] One important step, before using the triangular function, is to standardize the prototypes distances according to the distance of the first neighbor $(K+1)$ that is not taken into consideration when calculating the final prediction. This standardization is done, for each of the K-nearest neighbors, as follows:

$$d(\mathbf{x}, \mathbf{w}_k) = \frac{d(\mathbf{x}, \mathbf{w}_k)}{d(\mathbf{x}, \mathbf{w}_{K+1})}, \tag{6}$$

so that standardized distances always take values within the interval [0,1]. The WKNN algorithm can be summarized by the next steps:

1) Consider $H = \{(\mathbf{w}_q, \tilde{c}_q), q = 1, ..., Q\}$ a model composed by a set of prototypes $\mathbf{w}_q$ and their labels $\tilde{c}_q$, and $\mathbf{x}(t)$ a new observation whose class label has to be predicted.
2) Find the $K+1$ nearest neighbors $\{\mathbf{w}_k\}_{k=1}^{K+1}$ to $\mathbf{x}(t)$ according to a distance measure $d(\mathbf{w}_q, \mathbf{x}(t))$.
3) Use the $(K+1)$-th neighbor to standardize the K smallest distances via Eq. (6).
4) Use an arbitrary function $f(d(\cdot,\cdot))$ to transform the standardized distances into similarity measures.
5) As a prediction $\hat{c}_n$ for the observation's class, choose the class which shows the biggest similarity measure sum

$$\hat{c}_n = \max_{\forall c} \left(\sum_{k=1}^{K} f(d(\mathbf{w}_k, \mathbf{x}_n)).I(\tilde{c}_k, c) \right), \tag{7}$$

where the function $I(\tilde{c}_k, c)$ is either equal to 1, if $\tilde{c}_k = c$, or 0, otherwise.

Kernel methods can also be applied to the WKNN, giving rise to the KWKNN [19]. In this strategy, the kernelized squared Euclidean distance, specified at Eq. (4), can be used as a distance, so that

$$\begin{aligned} d(\mathbf{w}_q, \mathbf{x}(t)) &= \|\phi(\mathbf{x}(t)) - \phi(\mathbf{w}_q)\|_2^2 \\ &= \kappa(\mathbf{x}(t), \mathbf{x}(t)) + \kappa(\mathbf{w}_q, \mathbf{w}_q) - 2\kappa(\mathbf{x}(t), \mathbf{w}_q). \end{aligned} \tag{8}$$

For the proposed approach of the current paper, the KWKNN with the triangular function is used as classifier for new incoming test samples. With this method, the model's prediction is computed as

$$\hat{\mathbf{y}}(t) = \frac{\sum_{k=1}^{K} \mathbf{y}(\mathbf{w}_k).f(d(\mathbf{x}(t), \mathbf{w}_k))}{\sum_{k=1}^{K} f(d(\mathbf{x}(t), \mathbf{w}_k))}. \tag{9}$$

Similar to the one-hot encoding, one can simply get the highest value in the output prediction vector to define the sample's class estimation.

In the next section, the sparsification procedures used to build a dictionary by selecting relevant samples from a dataset are detailed.

[1] for other examples, see [17].

III. Sparsification Procedures

The training method of the proposed approach is based on sparsification procedures for the construction of a dictionary consisting of a subset from the training dataset. As these procedures are normally used in the field of KAF (i.e., signal processing), in this section we discuss how these procedures can be adapted for kernel prototype-based classifiers building.

A. Approximate Linear Dependence Criterion

This procedure, proposed by [5], automatically selects a subset of the training samples in order to construct a dictionary $\mathcal{D}_{t-1} = \{(\tilde{\mathbf{x}}_j, \tilde{c}_j)\}_{j=1}^{Q_{t-1}}$, whose size is determined by a single scalar parameter. The samples $(\tilde{\mathbf{x}}_j, \tilde{c}_j)$ in $\mathcal{D}_{t-1}$ are *approximately* linearly independent feature vectors.

Initially, the first training sample $(\mathbf{x}_1, c_1)$ is added to the dictionary. Then, at training time step t $(2 \leq t \leq N_{tr})$, after having observed $t-1$ training samples, the dictionary $\mathcal{D}_{t-1}$ is comprised of a relevant subset $\{(\tilde{\mathbf{x}}_j, \tilde{c}_j)\}_{j=1}^{Q_{t-1}}$ of these samples. When a new incoming training sample $(\mathbf{x}_t, c_t)$ is available, one must test whether it should be added or not to the dictionary. In order to do this, it is necessary to estimate a vector of coefficients $\mathbf{a} = \left(a_1, ..., a_{m_{t-1}}\right)^T$ satisfying the ALD criterion

$$\delta_1(t) \leq \nu_1, \tag{10}$$

where

$$\delta_1(t) \stackrel{def}{=} \min_{\mathbf{a}} \left\| \sum_{j=1}^{Q_{t-1}} a_j \phi\left(\tilde{\mathbf{x}}_j\right) - \phi\left(\mathbf{x}_t\right) \right\|^2, \tag{11}$$

and ν_1 is the sparsity level parameter (which is a hyperparameter and should be optimized). Developing the minimization problem in Eq. (11) and using the kernel trick, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$, one can write

$$\begin{aligned} \delta_1(t) \stackrel{\text{def}}{=} \min_{\mathbf{a}} \Bigg\{ & \sum_{i,j=1}^{Q_{t-1}} a_i a_j \kappa(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) - 2 \sum_{j=1}^{Q_{t-1}} a_j \kappa(\tilde{\mathbf{x}}_j, \mathbf{x}_t) \\ & + \kappa(\mathbf{x}_t, \mathbf{x}_t) \Bigg\} \end{aligned} \tag{12}$$

or, using the matrix notation,

$$\delta_1(t) = \min_{a} \left\{ \mathbf{a}^T \tilde{\mathbf{K}}_{t-1} \mathbf{a} - 2\mathbf{a}^T \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right) + k_{tt} \right\}, \tag{13}$$

where $\tilde{\mathbf{K}}_{t-1}$ is a $Q_{t-1} \times Q_{t-1}$ kernel matrix built using the current dictionary. The (i,j)-th entry of this matrix is given by $\left[\tilde{\mathbf{K}}_{t-1}\right]_{i,j} = \kappa\left(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j\right)$, with $i, j = 1, \ldots, Q_{t-1}$. The Q_{t-1}-dimensional vector $\tilde{\mathbf{k}}_{t-1}$ is defined as

$$\begin{aligned} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \big[& \kappa(\mathbf{x}_t, \tilde{\mathbf{x}}_1) \ \cdots \ \kappa(\mathbf{x}_t, \tilde{\mathbf{x}}_i) \ \cdots \\ & \kappa(\mathbf{x}_t, \tilde{\mathbf{x}}_{Q_{t-1}}) \big]^T, \end{aligned} \tag{14}$$

while $k_{tt} = \kappa\left(\mathbf{x}_t, \mathbf{x}_t\right)$. Solving Eq. (13) leads to the optimal $\mathbf{a}_t$, which is given by

$$\mathbf{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right), \tag{15}$$

so that Eq. (11) can be rewritten as

$$\delta_1(t) = k_{tt} - \tilde{\mathbf{k}}_{t-1}\left(\mathbf{x}_t\right)^T \mathbf{a}_t \tag{16}$$

or

$$\delta_1(t) = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t). \quad (17)$$

If $\delta_1(t) > \nu_1$, then the sample $(\mathbf{x}_t, c_t)$ must be added to the dictionary; that is, $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, c_t)\}$ and $Q_t = Q_{t-1} + 1$. However, if $\delta_1(t) \leq \nu_1$, the sample is approximately linearly dependent and must not be added to the dictionary; that is, $\mathcal{D}_t = \mathcal{D}_{t-1}$ and $Q_t = Q_{t-1}$.

B. Coherence Criterion

The authors in [8] proposed a sparsification strategy that employs a coherence parameter in order to control a model order increase. They applied this strategy for nonlinear filtering algorithms building in order to solve nonlinear dynamical systems identification.

They define the coherence parameter ψ as

$$\psi = \max_{i \neq j} |\kappa(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)| \; \forall i, j \in \mathcal{D} \quad (18)$$

where κ is a unit-norm kernel. This parameter reflects the most extreme correlations in a dictionary. Then, in time step t, they suggest inserting a new sample $(\mathbf{x}_t, c_t)$ to a dictionary $\mathcal{D}_{t-1}$ if the coherence of the increased dictionary remains below a given threshold ν_1, namely

$$\delta_1(t) = \max_{\forall \tilde{\mathbf{x}}_i \in \mathcal{D}_{t-1}} |\kappa(\tilde{\mathbf{x}}_i, \mathbf{x}_t)| \leq \nu_1 \quad (19)$$

where $\nu_1 \in [0, 1[$ determines both the sparsity level and the coherence of the dictionary. In summary, if $\delta_t \leq \nu_1$, then the sample $(\mathbf{x}_t, c_t)$ must be added to the dictionary; that is, $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, c_t)\}$ and $Q_t = Q_{t-1} + 1$. However, if $\delta_t > \nu_1$, the sample is extremely correlated to at least one of the current samples and must not be added to the dictionary, so that $\mathcal{D}_t = \mathcal{D}_{t-1}$ and $Q_t = Q_{t-1}$.

It is important to mention that, to calculate the coherence measure, one does not need to compute the kernel matrix. Also, the computational complexity of this criterion is only linear in the dictionary size. Finally, similarly to the ALD criterion, it uses only the inputs in order to decide if a sample must be added or not to the dictionary. The next two criteria use both input and output of a sample in order to do this.

C. Novelty Criterion

The author in [7] created a model named *resource-allocating network* (RAN), that allocates a new computational unit whenever an unusual pattern is presented to the network. This algorithm is used to predict complex time series, such as the chaotic time series created by the Mackey-Glass delay-difference equation. The RAN consists of a two-layer network, and uses a double-check novelty condition. An input-output pair $(\mathbf{x}_t, \mathbf{y}_t)$ is considered novel if the input is far away from existing centers (of the first layer), and if the difference between the desired output and the network output (the second layer output) is large. Here, these conditions were adapted, in order to build sparse kernel prototype-based classifiers, in the following manner.

Similarly to the other sparsification procedures, at time step t, a new sample $(\mathbf{x}_t, c_t)$ is presented to the model, which is composed by a dictionary $\mathcal{D}_{t-1} = \{(\tilde{\mathbf{x}}_j, \tilde{c}_j)\}_{j=1}^{Q_{t-1}}$. However, in order to be added to the dictionary, the new sample must fulfill two conditions. Firstly, the input of this sample must be far from the closest input of a prototype in the dictionary, considering some (dis)similarity measure such as

$$\delta_1(t) = \|\tilde{\mathbf{x}}_{j^*} - \mathbf{x}_t\|_2^2 > \nu_1, \quad (20)$$

where ν_1 as a hyperparameter and $\tilde{\mathbf{x}}_{j^*}$ is the closest prototype in the dictionary to the input sample. Here, the kernelized squared Euclidean measure

$$\begin{aligned} \delta_1(t) &= \|\phi(\tilde{\mathbf{x}}_{j^*}) - \phi(\mathbf{x}_t)\|_2^2 > \nu_1 \\ &= \kappa(\tilde{\mathbf{x}}_{j^*}, \tilde{\mathbf{x}}_{j^*}) + \kappa(\mathbf{x}_t, \mathbf{x}_t) - 2\kappa(\tilde{\mathbf{x}}_{j^*}, \mathbf{x}_t) > \nu_1 \end{aligned} \quad (21)$$

is used. The second condition is based on the difference between the desired output and the predicted one.

Firstly, one must calculate the predicted output $\hat{\mathbf{y}}_t$ using any KNN strategy. Then, if the difference between the real output $\mathbf{y}_t$ and the predicted output $\hat{\mathbf{y}}_t$ is greater than a hyperparameter ν_2, such as

$$\delta_2(t) = \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|_2^2 > \nu_2, \quad (22)$$

the sample must be added to the dictionary.

This second condition can also be extended to classification problems as follows: if the current model H_t makes an error:

$$\hat{c}_t \neq c_t, \quad (23)$$

in other words, if the predicted class $\hat{c}_t = \arg\max \hat{\mathbf{y}}_t$ differs from the actual class $c_t = \arg\max \mathbf{y}_t$, the second condition is satisfied and the sample must be added to the dictionary.

When comparing these two ways of applying the second condition, it was observed that the error-based approach as shown in Eq. (23) yields the best results. Therefore, this approach is utilized for comparing the novelty with the other sparsification criteria.

As a final remark, an important feature of the novelty criterion is that, similarly to the coherence, one does not need to compute a kernel matrix as in the ALD criterion.

D. Surprise Criterion

The authors in [6] introduced an information measure approach, called *Surprise*, in order to determine useful data to be learned and remove redundant ones. This criterion captures the amount of information a datum contains which is transferable to a learning system. The authors claim that methods like ALD, Coherence and Novelty are effective in practical applications, but are heuristic in nature. So, the surprise information criterion provides a unifying view on existing sparsification methods.

They define Surprise as the *Negative Log Likelihood* (NLL) of an observation, given the learning machine's hypothesis. Also, this measure is computed based on the theory of Gaussian Process Regression (GPR).

More formally, Surprise is a subjective information measure of a sample $(\mathbf{x}_t, c_t)$ with respect to a learning system H. Denoted by $S_H(\mathbf{x}_t, c_t)$, it is defined as the NLL of the

exemplar given the learning system's hypothesis on the data distribution

$$S_H(\mathbf{x}_t, c_t) = -\ln p(\mathbf{x}_t, c_t|H), \tag{24}$$

where $p(\mathbf{x}_t, c_t|H)$ is the subjective probability of $(\mathbf{x}_t, c_t)$ hypothesized by H. Applying this definition directly to the active online learning problem, we have the surprise of $(\mathbf{x}_t, c_t)$ to the current learning system H_{t-1}, such as

$$S_{H_{t-1}}(\mathbf{x}_t, c_t) = -\ln p(\mathbf{x}_t, c_t|H_{t-1}), \tag{25}$$

where $p(\mathbf{x}_t, c_t|H_{t-1})$ is the posterior distribution of $(\mathbf{x}_t, c_t)$ hypothesized by H_{t-1}. For simplicity, they define $S_t = S_{H_{t-1}}(\mathbf{x}_t, c_t)$.

If $p(\mathbf{x}_t, c_t|H_{t-1})$ is very large, the new datum $(\mathbf{x}_t, c_t)$ is well expected by the learning system H_{t-1}. If $p(\mathbf{x}_t, c_t|H_{t-1})$ is small, the new datum "surprises" the learning system, which means either the data contains something new for the system to discover or it is suspicious. According to this measure, we can classify the new exemplar into three categories:

- Redundant: $S_t < \nu_1$;
- Learnable: $\nu_1 \leq S_t \leq \nu_2$;
- Abnormal: $S_t > \nu_2$,

where ν_1 and ν_2 are hyperparameters. The choice of these thresholds are problem-dependent.

When developing this criterion with the GPR model, firstly, [6] calculated the output prediction as

$$\hat{y}_t = \tilde{\mathbf{k}}_{t-1}^T(\mathbf{x}_t)\left[\sigma_n^2\mathbf{I} + \tilde{\mathbf{K}}_{t-1}\right]^{-1}\mathbf{y}_{t-1}, \tag{26}$$

where $\tilde{\mathbf{k}}_{t-1}$ and $\tilde{\mathbf{K}}_{t-1}$ were already defined for the ALD criterion, σ_n^2 is the variance of the noise contained in an observation (here, we consider it a constant $\sigma_n^2 = 0.0001$), $\mathbf{I}$ is the identity matrix, and $\mathbf{y}_{t-1}$ is a vector containing all the prototypes outputs. Then, the authors calculate the prediction variance as

$$\sigma_t = \sigma_n^2 + k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T\left[\sigma_n^2\mathbf{I} + \tilde{\mathbf{K}}_{t-1}\right]^{-1}\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t). \tag{27}$$

It is important to mention that, if the noise variance σ_n^2 is considered 0, Eq. (27) is equal to Eq. (17) of the ALD method.

Then, discarding constant terms, considering just one output, and applying the definition of Surprise in Eq. (24), this measure can be written as

$$S_t = \ln\sigma_t + \frac{(y_t - \hat{y}_t)^2}{2\sigma_t^2}. \tag{28}$$

Some observations were made. First, little surprise is expected when there is redundancy in the data. Second, the proposed measure offers a general framework for redundancy removal, anomaly detection, and knowledge discovery. Third, the measure is proportional to both the magnitude of the prediction error and the prediction variance (particularly when the error is very small). Finally, the surprise measure becomes large when the prediction error is high and the variance is low.

Here, for prototype-based models, this measure was adapted to

$$S_t = \ln\sigma_t + \frac{\|\mathbf{y}_t - \hat{\mathbf{y}}_t\|_2^2}{2\sigma_t^2}, \tag{29}$$

where $\mathbf{y}_t$ and $\hat{\mathbf{y}}_t$ are, respectively, the sample's output and the prototype-based model's prediction, and σ_t is calculated using Eq. 27.

IV. The proposed Approach

As mentioned before, the dictionary of prototypes can be used as a standalone classifier by means of kernelized distances, such as those introduced in Section II. For this purpose, we follow the approaches suggested in [1], which are reproduced below for the sake of completeness.

Design Method 1: This approach involves the construction of a single dictionary. The first sample of the shuffled dataset will be the first item of the dictionary. Then, for each new sample, apply any of the aforementioned sparsification procedures. Note that each prototype vector in $\mathcal{D}_t$ carries its class label c_t for classification purposes.

Design Method 2: This approach requires the construction of smaller class-conditional dictionaries which are merged later. For a problem with C classes, it is required C dictionaries $\mathcal{D}_t^{(k)}$, $k = 1, 2, \ldots, C$. For this purpose, whenever a data sample $(\mathbf{x}_t, y_t)$ from the k-th class emerges, apply the Design Method 1 to the dictionary $\mathcal{D}_{t-1}^{(k)}$. The model's dictionary is the union of all the C class-conditional dictionaries $\mathcal{D}_t = \mathcal{D}_t^{(1)} \cup \mathcal{D}_t^{(2)} \cup \cdots \cup \mathcal{D}_t^{(C)}$.

We named the algorithm that combines the sparsification-based sample addition procedures and KWKNN classification as SPARse Kernel KWKNN (SPARK-KWKNN). One important feature of the training stage of this method is that it requires only a single pass over the training data, which contributes to its efficiency. In the next section, the datasets are briefly presented and the training and evaluation of the SPARK-KWKNN classifier are described.

V. Datasets and Methods

The motor failure dataset [20] consists of 294 attribute vectors, each one containing 6 harmonics of the fast Fourier Transform from a line current measurement of a three-phase induction motor. The samples are organized into 7 classes, where 1 is for normal operation condition (42 samples) and the other 6 correspond to short-circuit operation conditions (252 samples). In [20], the best results were reached when the problem was treated as a binary classification task and when the classes were balanced (adding 210 synthetic samples of normal condition to the dataset). So, the same methodology was used here.

In the Vertebral Column dataset[2] [21], six biomechanical attributes are derived from the shape and orientation of the pelvis and lumbar spine of 310 patients. The patient's condition was classified as Normal (100 samples), Disk Hernia (60 samples) or Spondylolisthesis (150 samples). This dataset can be also treated as a binary classification task, classifying the patient's conditions being normal (100) or abnormal (210).

The Pap-smear (Cervical Cancer) dataset [22] consists of 917 images of Pap-smear cells classified by cyto-technicians

[2]https://doi.org/10.24432/C5K89B

and doctors[3]. Each cell is described by 20 numerical features, and the cells fall into 7 classes. Samples from 3 classes originate from normal cells (totaling 242 samples) and samples from the other 4 classes correspond to abnormal cells (totaling 675 samples). So, the classification problem for this dataset is treated as a binary one in the current paper.

Finally, in the last dataset[4] [23], a wall-following robot chooses between four actions (MoveForward, Slight-Right-Turn, Sharp-Right-Turn, Slight-Left-Turn) as the robot navigates a room following the wall, using 24 ultrasound sensors arranged circularly around its body. The output of each ultrasound can be used as an attribute, or their outputs can be merged in 4 or 2 attributes. Here, the 2 attributes configuration is used.

In the experiments, for each evaluated dataset, we test 32 variants of the proposed algorithm, consisting of 4 different sparsification procedures (ALD, coherence, novelty and surprise), 2 design methods and 4 kernel functions (linear, gaussian, cauchy and log). Also, 100 independent realizations are executed. For each run, three steps are performed, namely: (i) holdout (shuffling and partitioning the data between training and test sets), (ii) training (hyperparameters optimization and parameters update), (iii) performance testing. For the holdout step, the data is randomly divided as follows: 70% for training and the remaining 30% for testing purposes. At the end of the testing phase, several statistical figures of merit for the performance of each classifier are computed (such as accuracy, error and F1-score). A 5-fold cross-validation strategy is performed in order to search the optimal values of the hyperparameters ν_1 and ν_2, those associated with each kernel function and K of KWKNN. The figure of merit for evaluating the performances of the algorithms while choosing the optimal hyperparameters is given by

$$J_{PB}(\mathcal{D}, E_b) = E_b + \lambda D_s, \tag{30}$$

where E_b is the classifier's batch error ($0 \leq E_b \leq 1$) already defined in Eq. (1), D_s ($0 \leq D_s \leq 1$) is the ratio between the number of samples selected to the dictionary Q and the number of training samples N_{tr}, and λ is a weighting term between these two factors ($\lambda \geq 0$). If $\lambda = 0$, the number of prototypes is not taken into account. By increasing λ, the J_{PB} index penalizes hyperparameters that lead the algorithm to build dictionaries with a large number of prototypes. In the current experiments, λ was set to 0.5.

It is worth noting that there are two restrictions on the hyperparameters choice. Firstly, the minimum number of samples selected to the dictionary must be greater than the number of classes of the problem. This restriction prevents hyperparameters' values that do not allow the addition of prototypes to the model. Also, the maximum number of samples selected to the dictionary, in the hyperparameter optimization procedure, must be less than 50% of the total number of training samples used. This prevents the choice of hyperparameters' values that allow excessive and, hence, unnecessary insertions of prototypes into the dictionary.

[3]http://mde-lab.aegean.gr/downloads

[4]https://doi.org/10.24432/C57C8W

TABLE II
MEAN ACCURACY FOR THE MOTOR FAILURE DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	62.5± 6.5	93.4± 1.8	97.3± 1.5	86.6± 14.5
	2	70.3± 4.7	93.5± 2.0	97.4± 1.4	91.4± 2.4
COH	1	90.9± 2.4	93.7± 1.9	98.8± 1.3	92.1± 2.2
	2	93.2± 1.9	80.7± 24.5	99.7± 0.7	92.0± 2.4
NOV	1	89.3± 3.0	**99.9± 0.7**	90.7± 2.8	90.3± 2.7
	2	89.7± 2.5	**99.9± 0.4**	91.4± 3.1	91.3± 2.6
SUR	1	61.8± 6.7	92.7± 2.0	96.2± 1.7	93.4± 1.6
	2	66.5± 5.6	88.3± 3.1	95.8± 1.9	93.2± 2.0

In the next section, we report the results of comprehensive computer simulations verifying the classification performance of the SPARK-KWKNN when applied to the aforementioned real-world datasets.

VI. RESULTS AND DISCUSSION

The results shown in this section are organized by each dataset. In the tables presented in the following subsections, *SM* and *DM* stand for, respectively, "Sparsification Method" and "Design Method". Also, for all the following experiments, the feature vectors are normalized to a zero mean and unit variance, and all models were implemented from scratch using the MATLAB software, version R2018a, running on Windows 11 Home, installed in a HP notebook with 2.70 GHz Intel Core i7 processor, 16 GB of RAM.

A. Motor Failure Dataset

The results of the SPARK-KWKNN classifier with the motor failure dataset are reported in Tables II and III. As can be seen, the best results were obtained using the novelty criterion and the Gaussian kernel. The mean accuracy achieved was 99.9% and it required only 12.5% of the training dataset samples (44 prototypes from 352 training samples). Also, for this dataset, the novelty sparsification criterion was the one that led to high accuracy rates with a low number of prototypes. Finally, comparing these results with other works, in [20], a similar mean accuracy was achieved by the LSSVM classifier, but it needed all the training samples as support vectors.

B. Vertebral Column Dataset

The mean accuracies of the SPARK-KWKNN classifier with the Vertebral Column dataset are reported in Table IV. The best results were achieved by using the coherence criterion and the log kernel. However, 173 prototypes (out of 217 training samples) were needed. A more balanced result was achieved by using the coherence criterion and the linear kernel (77.9%

TABLE III
NUMBER OF PROTOTYPES FOR THE MOTOR FAILURE DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	7	223	226	281
	2	21	221	220	281
COH	1	192	225	225	281
	2	188	220	214	281
NOV	1	65	**46**	79	77
	2	80	**44**	79	71
SUR	1	7	225	113	226
	2	14	190	146	242

TABLE IV
MEAN ACCURACY FOR THE VETEBRAL COLUMN DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	68.5± 7.7	73.2± 5.9	75.8± 5.5	79.8± 4.4
	2	67.5± 7.5	76.5± 5.9	74.6± 5.9	79.7± 6.6
COH	1	**77.9± 4.8**	74.7± 4.7	73.3± 6.2	**80.4± 3.2**
	2	65.5± 9.5	71.8± 5.4	69.5± 5.2	80.3± 3.7
NOV	1	74.9± 5.1	77.2± 5.5	76.5± 6.8	76.6± 5.2
	2	70.6± 5.4	76.3± 6.2	71.9± 6.6	76.1± 5.6
SUR	1	67.6± 9.2	72.0± 5.9	79.8± 3.5	76.4± 5.3
	2	68.4± 7.3	71.1± 6.2	70.9± 9.5	77.5± 4.4

mean accuracy with 39 prototypes). The ROC curve of this configuration is shown in Figure 1. As can be seen, the curve indicates that the classifier has a strong ability to distinguish between the two classes, as it remains close to the top-left corner of the plot. Comparing these results with other works, in [21], an accuracy of 85.9% was reported using a SVM with the KMOD kernel function.

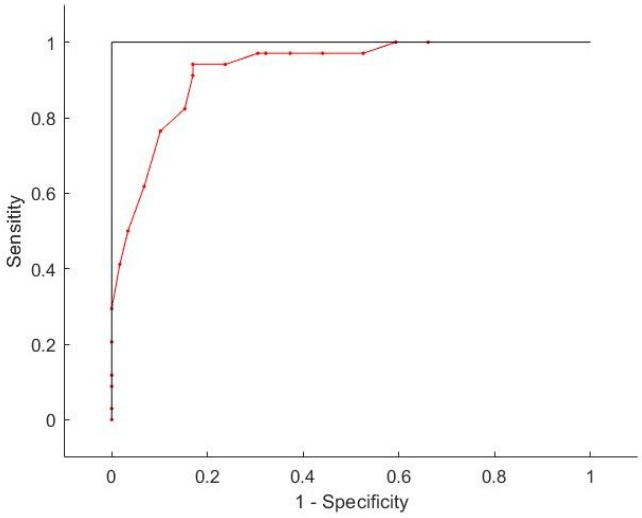

Fig. 1. ROC of Vertebral Column dataset and SPARK-KWNNN.

TABLE V
MEAN ACCURACY FOR THE PAP-SMEAR DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	87.6± 5.3	89.4± 6.1	83.1± 7.0	90.3± 12.1
	2	89.4± 6.4	85.6± 6.9	80.3± 15.4	**93.9± 2.0**
COH	1	90.4± 5.8	76.6± 20.8	87.9± 6.0	92.4± 1.1
	2	90.8± 5.0	87.1± 4.6	76.8± 8.7	90.3± 1.2
NOV	1	81.2± 8.1	86.2± 5.5	84.9± 2.6	87.1± 4.3
	2	87.3± 2.9	84.7± 2.6	84.4± 2.9	81.2± 18.3
SUR	1	92.2± 3.2	51.5± 37.6	91.9± 1.2	62.0± 35.6
	2	92.3± 2.6	90.4± 1.2	89.2± 4.4	76.1± 20.6

C. Pap-smear Dataset

The results of the SPARK-KWKNN classifier with the Pap-smear dataset are reported in Table V. The best mean accuracy of 93.9% was obtained using the ALD method and the log kernel. To achieve this performance, the model needed 18.85% of the training dataset (89 prototypes out of 475 samples). The ROC curve of this configuration is shown in Figure 2. As the curve remains close to the top-left corner of the plot, this indicates that the classifier has a strong ability to distinguish between the two classes. Comparing these results with other works, in [1], the mean accuracy values with all tested models were below 90%.

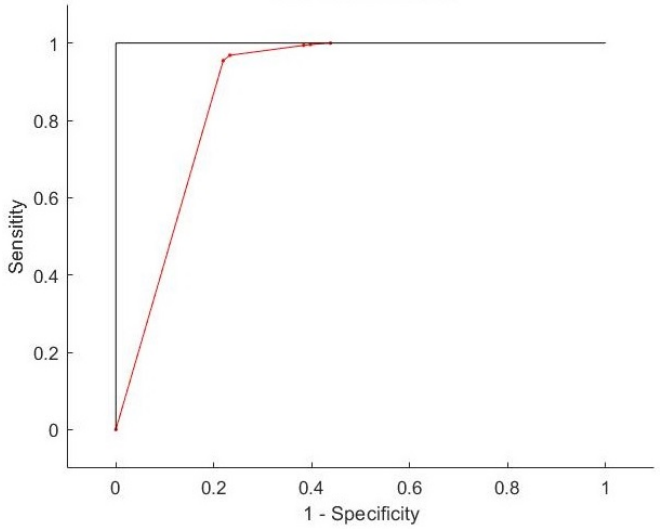

Fig. 2. ROC of Pap-smear dataset and SPARK-KWNNN.

D. Wall-following Dataset

The results of the SPARK-KWKNN classifier with the wall-following dataset are reported in Tables VI and VII. Using the surprise criterion and the Cauchy kernel, a mean accuracy of 96.3% was achieved, with only 3.2% of the training dataset

TABLE VI
MEAN ACCURACY FOR THE WALL FOLLOWING DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	45.3± 19.8	93.3± 1.1	93.9± 1.1	90.7± 19.8
	2	73.5± 5.8	95.3± 0.9	95.2± 1.0	96.0± 1.7
COH	1	93.5± 1.1	94.4± 0.9	93.8± 1.0	96.3± 0.5
	2	94.4± 0.8	95.2± 0.9	95.1± 1.0	96.3± 0.5
NOV	1	95.0± 0.9	95.1± 0.8	95.2± 0.9	95.3± 1.1
	2	75.3± 7.9	63.0± 9.2	64.2± 9.5	95.3± 0.9
SUR	1	50.1± 17.0	96.3± 0.5	**96.3± 0.5**	96.4± 0.5
	2	79.7± 6.2	96.1± 0.7	96.5± 0.4	96.4± 0.6

TABLE VII
NUMBER OF PROTOTYPES FOR THE WALL FOLLOWING DATASET

SM	DM	linear	gauss	cauchy	log
ALD	1	6	159	174	732
	2	12	162	137	732
COH	1	139	158	174	732
	2	151	125	152	732
NOV	1	64	74	81	75
	2	15	4	4	72
SUR	1	11	171	**124**	229
	2	26	132	634	190

(124 prototypes out of 3819 samples). Comparing these results with other works, in [23], a mean accuracy of 96.8% was obtained using a MLP of one hidden layer.

VII. CONCLUSION

In this work, a new framework called SPARK-KWKNN was proposed for designing sparse kernel prototype-based classifiers. The experimental results demonstrated that the algorithm can achieve high mean accuracy while maintaining sparse models with a low number of prototypes.

An important observation is that increasing the number of prototypes does not necessarily lead to improved classification performance. In many cases, the best results were obtained with configurations that selected significantly fewer prototypes, emphasizing the importance of effective sparsification over model complexity.

Furthermore, no single sparsification method or kernel function consistently outperformed the others across all datasets. These findings highlight that the effectiveness of this model depends on the characteristics of the dataset, reinforcing the need for careful selection of sparsification criteria, kernel functions, and hyperparameters tailored to each specific task.

REFERENCES

[1] D. N. Coelho and G. A. Barreto, "Approximate linear dependence as a design method for kernel prototype-based classifiers," in *Advances in Self-Organizing Maps, Learning Vector Quantization, Clustering and Data Visualization (WSOM'2019)* (A. C. M. G. J. Vellido A., Gibert K., ed.), vol. 976, pp. 241–250, Springer, 2019.

[2] H. Yin, "On the equivalence between kernel self-organising maps and self-organising mixture density networks," *Neural Networks*, vol. 19, no. 6, pp. 780–784, 2006.

[3] J. Valyon, *Extended LSSVM for System Modeling*. PhD thesis, Budapest University of Technology and Economics, Department of Measurement and Information Systems, 2006.

[4] W. Liu, P. P. Pokharel, and J. C. Principe, "The kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, 2008.

[5] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[6] W. Liu, I. Park, and J. C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE transactions on neural networks*, vol. 20, no. 12, pp. 1950–1961, 2009.

[7] J. Platt, "A resource-allocating network for function interpolation," *Neural computation*, vol. 3, no. 2, pp. 213–225, 1991.

[8] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2008.

[9] K. W. Lau, H. Yin, and S. Hubbard, "Kernel self-organising maps for classification," *Neurocomputing*, vol. 69, no. 16, pp. 2033–2040, 2006.

[10] D. Hofmann and B. Hammer, "Sparse approximations for kernel learning vector quantization," in *Proceedings of the ESANN'2013*, pp. 549–554, 2013.

[11] L. A. Soares Filho and G. A. Barreto, "On the efficient design of a prototype-based classifier using differential evolution," in *2014 IEEE Symposium on Differential Evolution (SDE)*, pp. 1–8, IEEE, 2014.

[12] R. F. Albuquerque, P. D. de Oliveira, and A. P. d. S. Braga, "Adaptive fuzzy learning vector quantization (AFLVQ) for time series classification," in *North American Fuzzy Information Processing Society Annual Conference (NAFIPS'2018)*, pp. 385–397, Springer, 2018.

[13] M. Biehl, B. Hammer, and T. Villmann, "Prototype-based models in machine learning," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 7, no. 2, pp. 92–111, 2016.

[14] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558–569, 1993.

[15] T. Kohonen, "Improved versions of learning vector quantization," in *Proceedings of the 1990 International Joint Conference on Neural Networks (IJCNN'90)*, pp. 545–550, IEEE, 1990.

[16] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[17] K. Hechenbichler and K. Schliep, "Weighted k-nearest-neighbor techniques and ordinal classification," 2004.

[18] S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 4, pp. 325–327, 1976.

[19] G. Rubio, L. J. Herrera, H. Pomares, I. Rojas, and A. Guillén, "Design of specific-to-problem kernels and use of kernel weighted k-nearest neighbours for time series modelling," *Neurocomputing*, vol. 73, no. 10-12, pp. 1965–1975, 2010.

[20] D. N. Coelho, G. Barreto, C. M. Medeiros, and J. D. A. Santos, "Performance comparison of classifiers in the detection of short circuit incipient fault in a three-phase induction motor," in *Computational Intelligence for Engineering Solutions (CIES), 2014 IEEE Symposium on*, pp. 42–48, IEEE, 2014.

[21] A. R. da Rocha Neto, R. Sousa, G. de A. Barreto, and J. S. Cardoso, "Diagnostic of pathology on the vertebral column with embedded reject option," in *Pattern Recognition and Image Analysis: 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings 5*, pp. 588–595, Springer, 2011.

[22] J. Jantzen, J. Norup, G. Dounias, and B. Bjerregaard, "Pap-smear benchmark data for pattern classification," *Nature inspired Smart Information Systems (NiSIS 2005)*, pp. 1–9, 2005.

[23] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in *2009 6th Latin American robotics symposium (LARS 2009)*, pp. 1–6, IEEE, 2009.