RAIRO-Oper. Res. 57 (2023) 2493–2517
https://doi.org/10.1051/ro/2023114

RAIRO Operations Research
www.rairo-ro.org

OPT-RNN-DBFSVM: OPTIMAL RECURRENT NEURAL NETWORK DENSITY BASED FUZZY SUPPORT VECTOR MACHINE

Karim El Moutaouakil* and Abdellatif El Ouissari

Abstract. Two major problems are encountered when using fuzzy SVM: (a) the number of local minima increases exponentially with the number of samples and (b) the quantity of required computer storage, required for a regular quadratic programming solver, increases by an exponential magnitude as the problem size expands. The Kernel-Adatron family of algorithms gaining attention lately which has allowed to handle very large classification and regression problems. However, these methods treat different types of samples (Noise, border, and core) with the same manner, which causes searches in unpromising areas and increases the number of iterations. In this work, we introduce a hybrid method to overcome these shortcoming, namely Optimal Recurrent Neural Network Density Based fuzzy Support Vector Machine (Opt-RNN-DBFSVM). This method consists of four steps: (a) characterization of different samples, (b) elimination of samples with a low probability of being a support vector, (c) construction of an appropriate recurrent neural network based on an original energy function, and (d) solution of the system of differential equations, managing the dynamics of the RNN, using the Euler–Cauchy method involving an optimal time step. Thanks to its recurrent architecture, the RNN remembers the regions explored during the search process. We demonstrated that RNN-FSVM converges to feasible support vectors and Opt-RNN-DBFSVM has a very low time complexity compared to RNN-FSVM with constant time step, and KAs-FSVM. Several experiments were performed on academic data sets. We used several classification performance measures to compare Opt-RNN-DBFSVM to different classification methods and the results obtained show the good performance of the proposed method.

Mathematics Subject Classification. 90C20, 90C29, 90C90, 93E20.

Received October 5, 2022. Accepted July 28, 2023.

Introduction

The classification problem is an NP-Hard problem that has many applications in medicine, industry, economics and other fields. Several types of classifiers have been proposed in the literature to solve this problem, including the approach called Support Vector Machine based on Quadratic Programming (QP) [20,32,45]. The difficulty in the implementation of SVMs on massive data comes from the fact that the quantity of required computer storage required for a regular QP solver increases by an exponential magnitude as the problem size expands.

Keywords. Recurrent Neural Network (RNN), Fuzzy Support Vector Machine (FSVM), Kernel-Adatron algorithm (KA), Euler–Cauchy Algorithm.

Sidi Mohamed Ben Abdellah of Fez, Polydisciplinary Faculty of Taza, Laboratory of engineering science, Morocco.
*Corresponding author: yassirkarimimane@gmail.com

Since 2002 when the classic fuzzy SVM was developed [34], different studies were proposed to improve the efficiency of this fuzzy version, in combination with other algorithms and algorithms in different fields and domains, for example, DB-FSVM [17], Twin FSVM [39], FSVM-CIL [5], New fuzzy SVM [52], Improved FSVM [10], and others. In our case, we introduce the new Density based fuzzy support vector machine called, Optimal recurrent neural network density-based fuzzy support vector machine, which takes into account the importance of the optimal time size in each iteration of the Euler–Cauchy method.

SVMs approaches are based on the existence of a linear separator which can be obtained by exploding the data in a higher dimensional space through appropriate kernel functions. Among all possible hyperplanes, the SVM searches for the one with the most confident separation margin for a good generalization. This issue takes the form of a nonlinear constrained optimization problem that is usually handled using optimization methods. Thanks to the Kuhen–Tuker conditions, all these methods past to the dual versions and call the optimizations methods to find the support vectors on which the optimal margin is built. Unfortunately, the complexity in time and memory grow exponentially with the size of data sets; in addition, the number of local minima grow too, which influences the location of the separation margin and the quality of the predictions.

A primary area of research in the area of learning from empirical data through support vector machines (SVMs) and addressing classification and regression issues is the development of incremental learning designs when the size of the training dataset is massive [16]. Out of many possible candidates, avoiding the usage of regular Quadratic Programming (QP) solvers, the two learning methods gaining attention lately are Iterative Single-Data Algorithms (ISDA) and Sequential Minimal Optimization (SMO) [26,29,38,51]. ISDAs operate from a single sample at hand (pattern-based learning) towards the best fit solution. The Kernel AdaTron (KA) was the primary ISDA for SVMs, using kernel functions to mapping data to the high dimensional character space of SVMs [19] and conducting AdaTron [4] processing in the character space. Platt's SMO algorithm is an outlier among the so-called decomposition approaches introduced in [27,37], operating on a 2-samples workset of samples at a time. Because the decision for the two-point workset may be determined analytically, the SMO does not require the involvement of standard QP solvers. Due to it being analytically driven, the SMO has been especially wildly popular and is the most commonly utilized, analyzed, and further developed approach. Meanwhile, KA, though yielding somewhat comparable performance (accuracy and computational required time) in resolving classification issues, has not gained as much traction. The reason for this is twofold. First, up till lately [50], KA appeared to be restricted to classification tasks and second, it "missed" the flower of the robust theoretical framework. KA employs a gradient ascent procedure, and that fact also may have caused some researchers to be suspicious of the challenges posed by gradient ascent techniques in the presence of a perhaps ill-conditioned core array. In [30], for a lacking bias parameter b, the authors derive and demonstrate the equality of two apparently dissimilar ISDAs, namely a KA approach and an unbiased variant of the SMO training scheme [51] when constructing SVMs possessing positive definite kernels. The equivalence is applicable to the classification and regression tasks, and sheds additional insight in these apparently dissimilar methods of learning. Despite the richness of the toolbox set up to solve the quadratic programs from SVM, and with the large amount of data generated by social networks, medical and agricultural fields, etc., the amount of computer memory required for a QP solver from the SVM dual grows hyper-exponentially and additional methods implementing different techniques and strategies are more than necessary.

Classical algorithms, namely ISDAs and SMO, do not distinguish between different types of samples (noise, border, and core) which causes searches in unpromising areas. In this work, we introduce a hybrid method to overcome these shortcoming, namely Optimal Recurrent Neural Network Density Based Support Vector Machine (Opt-RNN-DBSVM). This method proceeds in four steps: (a) characterization of different samples based on the density of the data sets (noise, core, and border), (b) elimination of samples with a low probability of being a support vector, namely core samples that are very far from the borders of different components of different classes, (c) construction of an appropriate recurrent neural network based on an original energy function making balance between the SVM-dual components (constraints and objective function) and insuring the feasibility of the network equilibrium points [22,25], and (d) solution of the system of differential equations, managing the dynamics of the RNN, using the Euler–Cauchy method involving an optimal time step. Due to its

recurrent nature, the RNN was able to memorize locations visited during previous explorations. At one hand, two main intersting fondamental results were demonstrated: the convergence of RNN-SVM to faisible solutions and Opt-RNN-DBSVM has a very low time complexity compared to Const-RNN-SVM, SMO-SVM, ISDA-SVM, and L1QP-SVM. At the other hand, several experimental study were conducted based on well-known data sets. Based on well-known performance measures (Accuracy F1-score Precision Recal), Opt-RNN-DBSVM out performed recurrent neural network-SVM with constant time step, Kernel-Adatron family of algorithms-SVM familly, and well-known non-kernel models; In fact, Opt-RNN-DBSVM improved the accuracy, the F1Score, the precision, and the recall. Moreover, the proposed method requires a very small number of support vectors.

The rest of this paper is organized as follows: In the second section, we give the flowchart of the proposed method. In the third section, we give the outline of our recent SVM versions called Density Based Support Vector Machine. The fourth section presents, in detail, the construction of recurrent neural network associated with SVM-dual and the Euler–Cauchy algorithm that implements an optimal time step. At the fifth section, we give some experimental results. Finally, we give some conclusions and future extension of Opt-RNN-DBSVM.

1. The architecture of the proposed method

The Kernel AdaTron (KA) algorithms, namely ISDAs and SMO, treat different types of samples (noise, border, and core) with the same manner (all samples are considered for several iterations and supposed to be a support candidate with uniform probability), which causes searches in unpromising areas and increase the number of iterations. In this work, we introduce an economic method to overcome these shortcoming, namely Optimal Recurrent Neural Network Density Based Support Vector Machine (Opt-RNN-DBSVM). This method proceeds in four steps (see Fig. 1):

(1) Characterization of different samples based on the density of the data sets (noise, core, and border); to this end, two parameters are introduced: the size of the neighborhood of the current sample and the threshold that permits a such categorisation;
(2) Elimination of samples with a low probability of being a support vector, namely core samples that are very far from the borders of different components of different classes and the noise samples that contain a wrong information about the phenomenon under study. In our previews work [17], we demonstrate that a such suppression do not influence the performance of the classifiers;
(3) Construction of an appropriate recurrent neural network based on an original energy function making balance between the SVM-dual components (constraints and objective function) and ensuring the feasibility of the network equilibrium points [22, 25];
(4) Solution of the system of differential equations, managing the dynamics of the RNN, using the Euler–Cauchy method involving an optimal time step. In this regard, the formula of the coming state of the neurons, of the constructed RNN, is introduced into the energy function, which leads to a 1-dimension quadratic optimization problem whose solution represents the optimal step of the Euler–Cauchy process that ensures a maximum decrease of the energy function [15]. The components of the produced equilibrium point represent the membership degrees of different samples to the support vectors data set.

2. Density based support vector machine

In the following, we denote by BD the set of N samples $x_1, \ldots, x_N$ labeled, respectively, by $y_1, \ldots, y_N$, distributed *via* K class $C_1, \ldots, C_K$. In our case, $K = 2$ and $y_i \in \{-1, +1\}$.

2.1. Classical support vector machine

The hyperplane we are looking for must satisfy the equation $w.x_i + b = 0$, where w is the weight which defines this SVM separator that satisfies the constraints family given by $\forall i = 1, \ldots, N\ y_i(x_i.w + b) \geq 1$. To ensure a maximum margin, you need to maximize $\frac{2}{\|w\|}$. As the pattern is not linearly separable, we introduce the kernel functions K (that satisfy the Mercer conditions [35]) to explore the data in appropriate space.

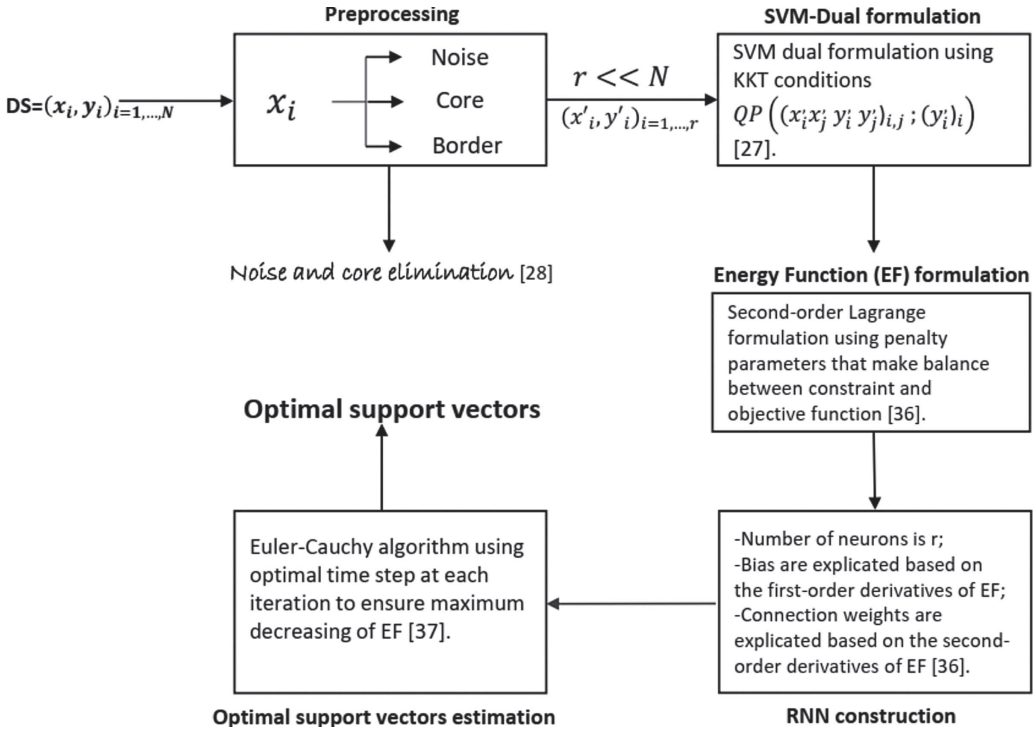


FIGURE 1. Opt_RNN_DBSVM flowchart.

By introducing the Lagrange relaxation and writing the Kuhn-Tuker conditions, we obtain a quadratic optimization problem with a single linear constraint that we solve to determine the support vectors [43].

To address the problem of sutured constraints, some researchers have added the notion of a soft margin [9]. They employed N supplementary slack variables $\xi_i \geq 0$ at every constraint $y_i(x_i.w + b) \geq 1$. The sum of the relaxed variables is weighted and included in the cost function:

$$\begin{cases} \text{Min}\dfrac{1}{2}\|w\|^2 + C\displaystyle\sum_{i=1}^{N}\xi_i \\ \quad \texttt{Subject to}: \\ \qquad y_i(\phi(x_i)\cdot w + b) \geq 1 - \xi_i \\ \qquad \xi_i \geq 0, \qquad \forall i = 1, \ldots, N. \end{cases}$$

Here ϕ represents the transformation function derived from the function kernel K. We obtain the coming dual problem:

$$\begin{cases} \text{Max}\displaystyle\sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i, x_j) \\ \quad \texttt{Subject to}: \\ \qquad \displaystyle\sum_{i=1}^{N}\alpha_i y_i = 0 \\ \qquad 0 \leq \alpha_i \leq C, \qquad \forall i = 1, \ldots, N. \end{cases}$$

A Fuzzy Support Vector Machine (FSVM) is a variant of an SVM and is proposed specifically for dealing with outliers and noise [34]. In an FSVM, samples are assigned different fuzzy membership values for characterizing their importance. Strategies for calculating fuzzy membership values are critical for the performance of an FSVM method. The fuzzy membership values of outliers and noise are normally relatively lower than the other samples. To determine fuzzy membership, we look for the main properties of the data set, and we relate those properties to the fuzzy membership, for example, Chun [34] assumes that time is the main property of the dataset, the latter gives the value m_i as a function of time and expresses the value of m_i as follows:

$$m_i = f(t_i)$$

with $0 < \sigma \leq m_i \leq 1$ and $t_1 \leq t_2 \leq \ldots \leq t_N$.

Chun [34] proposes to take that $s_1 = f(t_1) = \sigma$ and $s_N = f(t_N) = 1$ and also proposes two types of fuzzy value, the first one is a linear function type value which gave by:

$$m_i = f(t_i) = at_i + b$$

and we use board conditions to determine a and b, one finds:

$$m_i = f(t_i) = \frac{1-\sigma}{t_N - t_1} t_i + \frac{t_N \sigma - t_1}{t_N - t_1}$$

and the second is a quadratic function type value which has given by:

$$m_i = f(t_i) = a(t_i - b)^2 + c$$

and we use board conditions to determine a, b and c:

$$m_i = f(t_i) = (1-\sigma)\left(\frac{t_i - t_1}{t_N - t_1}\right)^2 + \sigma.$$

In [49] the authors used the class center method to generate the fuzzy membership. They denote x^+ and r^+ as the mean and radius of class $+1$ and x^- and r^- as the mean and radius of class -1, respectively. The radius of each class is the farthest distance between its training points and its class center, namely $r^+ = \max_{x_i, y_i = +1} \|x^+ - x_i\|$ and $r_- = \max_{x_i, y_i = -1} \|x^- - x_i\|$. For any training point x_i, its fuzzy membership m_i is defined as follows:

$$m_i = \begin{cases} 1 - \frac{\|x^+ - x_i\|}{r^+ + d}, & \text{if } y_i = +1 \\ 1 - \frac{\|x^- - x_i\|}{r^- + d}, & \text{if } y_i = -1 \end{cases} \tag{1}$$

where $d > 0$ is used to avoid the case $m_i = 0$.

Two new membership functions also were proposed by Dhanasekaran and Murugesan [10].

First new membership function ($\mathbf{MF_1}$). The overall center of data points is used. By considering the overall centroid, we have the advantage of reducing the membership value considerably for outliers. By doing so, we have considerably improved the performance of the model. Further, the point created with the maximum value of the coordinate and the point created with the minimum value of the coordinate from the overall data set is considered for the calculation of membership values. The distance between the point created with the maximum value of the coordinate and the point created with the minimum value of the coordinate is considered the d-value. The new membership function is:

$$m_i = 1 - \left(\frac{|X_c - X_i|}{d_1 + \Delta}\right), \quad \forall i.$$

If the data consist of p dimensions, then the point created with the maximum value of the coordinate is $X_{\max}$, where $X_{\max} = (\mathrm{Max}(x_1), \mathrm{Max}(x_2) \ldots \mathrm{Max}(x_p))$ and the point created with the minimum value of the coordinate

is $X_{\min}$, where $X_{\min} = (\text{Min}(x_1), \text{Min}(x_2) \dots \text{Min}(x_p))$. X_c is centroid coordinate and X_i represents coordinates. Where $\delta = 0.01$, which is the minimum value to make the denominator not become zero. Where d is the distance between $X_{\max}$ and $X_{\min}$.

$$d_1 = \sqrt{[\text{Max}(x_1) - \text{Min}(x_1)]^2 + [\text{Max}(x_2) - \text{Min}(x_2)]^2 + \ldots + [\text{Max}(x_p) - \text{Min}(x_p)]^2}.$$

Second new membership function ($\mathbf{MF_2}$). In the Second new membership function, the computation of the d value differs from the first new membership function. In this approach, the point created with the maximum value of the coordinate in the data set and the overall centroid is considered to calculate the membership values. The distance between the point created with the maximum value of the coordinate and the overall centroid of data points is considered as d-value.

$$m_i = 1 - \left(\frac{|X_c - X_i|}{d_2 + \Delta} \right)$$

d is the distance between $X_{\max}$ and X_c.

$$d_2 = \sqrt{[\text{Max}(x_1) - x_1']^2 + [\text{Max}(x_2) - x_2']^2 + \ldots + [\text{Max}(x_p) - x_p']^2}$$

Several methods can be used to solve this optimization problem: gradient methods, linearization method Frank-Wolf method, generation column method, Newton method applied to the Kuhn system [36], sub-gradient methods, Dantzig algorithm, Uzawa algorithm [36], recursive neural networks [18], hill climbing, simulated annealing, search by calf, A^*, genetic algorithm, ant colony [40], and particle swarm method [18] ... etc.

Since its appearance until today, different versions of SVM have been developed to improve its accuracy. There are several works in this direction [*], we mention, Density-based support vector machine [*], Density-based fuzzy support vector machine [*], Pinball Loss Support Vector Machine [48],[41], stochastic gradient Twin support vector machines (SG-TSVM) [34*], Least Squares Support Vector Machine Classifiers (LS-SVM) [9], generalized support vector machines (G-SVM) [32], Fuzzy support vector machines [20], One class Support Vector Machine (OC-SVM) [1], [35], Total Support Vector Machine (T-SVM) [2], Weighted Support Vector Machine (W-SVM) [34], Granular Support Vector Machines (G-SVM) [49], Smooth Support Vector Machine (S-SVM) [10], Proximity Support Vector Machine classifiers (P-SVM) [42], Multisurface proximal support vector machine classification *via* generalized eigenvalues (GEP-SVM) [6], and Twin support vector machines (T-SVM) [23] ... etc.

2.2. Density based fuzzy support vector machine (DB-FSVM)

In this section, we give a short description of DB-FSVM method. We introduce the actual number $r > 0$ and the integer $mp > 0$, called min-points, and we define three types of samples: noise point, border point, and interior point (or cord point). We showed that the interior points do not change their nature even when they are projected into another space by the kernel functions. Furthermore, we have established that such points cannot be selected as support vectors [17].

Definition 1. Let $S \subseteq \mathbb{R}^n$. A point $a \in \mathbb{R}^n$ is said to be an Interior Point (or cord point) of S if there exists an $r > 0$ such that $B(a, r) \subseteq S$. The set of all interior points of S is denoted by $\text{int}(S)$ or $\overset{o}{S}$.

Definition 2. For a given dataset BD, a non-negative real r and an integer mp, there exist three kind of samples.

(1) A sample x is called C_i-Noise Point (NP_i) if $|C_i \cap B(x, r)| < mp$.

(2) A sample x is called C_i-Cord Point (CP_i) if $|C_i \cap B(x, r)| \geq mp$ and $x \in \overbrace{\text{envol}(C_i)}^{o}$.

(3) A sample x is called C_i-Border Point (BP_i) if $|C_i \cap B(x, r)| < mp$ and there exists a C_i-cord point y such as $x \in B(y, r)$.

Let K be a kernel function allowing to move from the space $\mathbb{R}^n$ to the space $\mathbb{R}^N$ using the transformation ϕ (here $n < N$).

Lemma 3 ([17]). *If a is a C_i-Cord point for a given ϵ and minpoints (mp), then $\phi(a)$ is also a C_i-Cord point with appropriate ϵ' and the same minpoints (mp).*

Theorem 4 ([17]). *A cord-point is either a noise-point or an Border-point.*

Proposition 5 ([17]). *Lets $\epsilon > 0$ be a real number. The cord point set, cordPoints(minPoints), is decreasing function for the inclusion operator.*

Let $\{\alpha_1, \ldots, \alpha_n\} = \mathrm{BM} \cup \mathrm{CM} \cup \mathrm{NM}$ be the set of the Lagrange multipliers where BM, CM, and NM are the Lagrange multipliers of the border samples, cord samples, and noise samples, respectively.

As the elements of NM and CM can not selected to be support vector, the reduced dual problem is given by:

$$(\mathrm{RD}) \begin{cases} \mathrm{Max} \displaystyle\sum_{\alpha_i \in \mathrm{BM}} \alpha_i - \frac{1}{2} \sum_{\alpha_i \in \mathrm{BM}} \sum_{\alpha_j \in \mathrm{BM}} \alpha_i \alpha_j y_i y_j K(x_i x_j) \\ \quad \texttt{Subject to :} \\ \quad \displaystyle\sum_{\alpha_i \in \mathrm{BM}} \alpha_i y_i = 0 \\ \quad 0 \leq \alpha_i \leq m_i C \qquad \forall \alpha_i \in \mathrm{BM} \;\; \forall i = 1, \ldots, N. \end{cases}$$

In this work, as the (RD) problem is quadratic with linear constraint, we use the continuous Hopfield Network by proposing an original energy function in the coming section [12].

3. Recurrent neural network to optimal support vectors

The continuous Hopfield networks consist of interconnected neurons with a smooth sigmoid activation function (usually a hyperbolic tangent). The differential equation which governs the dynamics of the CHN is:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{u}{\tau} + W \cdot \alpha + I \tag{2}$$

where u, α, W and I are, respectively, the vectors of neuron states, the outputs, the weight matrix and the biases. For a CHN of N neurons, the state u_i and output α_i of the neuron i are relied by the equation $\alpha_i = \tanh(u_i) = f(u_i)$.

For an initial vector state $u^0 \in \mathbb{R}^N$, a vector $u^e \in \mathbb{R}^N$ is called an equilibrium point of the system (2), if and only if, $\exists t^e \in \mathbb{R}^+$ such as $\forall t \geq t^e \; u(t) = u^e$. It should be noted if the energy function (or Layapunov function) exists; the equilibrium point exists as well. Hopfield proved that the symmetry of matrix of the weight is a sufficient condition for the existence of Lyapunov function [24].

3.1. Continuous Hopfield network based on original energy function

To solve the obtained dual problem *via* recurrent neural network [12], we propose the following energy function:

$$E(\alpha_1, \ldots, \alpha'_N) = \beta_0 \sum_{\alpha_i \in D} \alpha_i - \frac{\beta_0}{2} \sum_{\alpha_i \in \mathrm{BM}} \sum_{\alpha_j \in \mathrm{BM}} \alpha_i \alpha_j y_i y_j K(x_i x_j) + \beta_1 \sum_{\alpha_i \in \mathrm{BD}} \alpha_i y_i + \frac{\beta_2}{2} \left(\sum_{\alpha_i \in \mathrm{BD}} \alpha_i y_i \right)^2.$$

To determine the vector of the neurons biases, we calculate the partial derivatives of E:

$$\frac{\partial E}{\partial \alpha_i}(\alpha_1, \ldots, \alpha'_N) = \beta_0 - \beta_0 \sum_{\alpha_j \in \mathrm{BM}} \alpha_j y_i y_j K(x_i x_j) + \beta_1 y_i + \beta_2 y_i \sum_{\alpha_j \in \mathrm{BD}} \alpha_j y_j.$$

The components of the biases vector are given by:

$$I_i = \frac{\partial E}{\partial \alpha_i}(0) = -\beta_0 - \beta_1 y_i \qquad \forall i = 1, \ldots, N'.$$

To determine the connection weights W between each neurons pair, we calculate the second partial derivatives of E:

$$\frac{\partial^2 E}{\partial \alpha_j \partial \alpha_i}(\alpha_1, \ldots, \alpha_{N'}) = -\beta_0 y_i y_j K(x_i, x_j) + \beta_2 y_i y_j.$$

The components of the weights W matrix are given by:

$$W_{i,j} = \frac{\partial^2 E}{\partial \alpha_j \partial \alpha_i}(0) = \beta_0 y_i y_j K(x_i, x_j) - \beta_2 y_i y_j.$$

To calculate the equilibrium point of the proposed recurrent neural network, we use the Euler–Cauchy iterative method:

(1) Initialization: $\alpha_1^0, \ldots, \alpha_{N'}^0$ and the step ρ^0 are randomly chosen;
(2) Given $\alpha_1^t, \ldots, \alpha_{N'}^t$ and the step ρ^t, the step ρ^{t+1} is chosen such that E^{t+1} is maximum and we calculate $u_1, \ldots, u_{N'}$ using: $\forall i = 1, \ldots, N', u_i^{t+1} = u_i^t + \rho^{t+1}(\sum_{j=1,\ldots,N'} W_{i,j}\alpha_j^t + I_i)$.
Then we calculate $\gamma_1, \ldots, \gamma_{N'}$ using the activation function f: $\gamma_i = f(u_i^{t+1})$.
Then the $\alpha_1^{t+1}, \ldots, \alpha_{N'}^{t+1}$ are given by: $\alpha^{t+1} = P(\gamma)$, where P is the projection operator on the set $\{\alpha \in \mathbb{R}^{N'} / \sum_{i=1}^{N'} \alpha_i y_i = 0\}$.
(3) Return to (1) until $\|\alpha^{t+1} - \alpha^t\| \leq \epsilon$, where $0 < \epsilon$ the Figure 2 shows the connection weights W between each pair of neurons.

Theorem 6. *If* $i = j, P_{i,j} = 1 - \frac{y_i^2}{N'}$, *else* $P_{i,j} = -\frac{y_i y_j}{N'}$, *where* $S = \sum_{i=1,\ldots,N'} y_i^2$.

Proof. We have $P = I - A^t \cdot (A \cdot A^t)^{-1} \cdot A$ and $A = [y_1, \ldots, y_{N'}]$.
Then $(A \cdot A^t)^{-1} = ([y_1, \ldots, y_{N'}] \cdot [y_1, \ldots, y_{N'}]^t)^{-1} = \frac{1}{N'}$ because $N' = \sum_{i=1,\ldots,N'} y_i^2$.
Thus $A^t \cdot (A \cdot A^t)^{-1} \cdot A = \frac{1}{N'} \cdot [y_1, \ldots, y_{N'}]^t \cdot [y_1, \ldots, y_{N'}]$.
Finely, for $i = j$, $P_{i,j} = 1 - \frac{y_i^2}{N'}$, and for $i \neq j, P_{i,j} = -\frac{y_i \cdot y_j}{N'}$.
Concerning the constraint family satisfaction $0 \leq \alpha_i \leq m_i C, \forall i = 1, \ldots, N$, we use the activation function:

$$f(x) = m_i C \cdot \tanh\left(\frac{x}{\tau}\right) = m_i C \cdot \frac{e^{x/\tau} - e^{-x/\tau}}{e^{x/\tau} + e^{-x/\tau}},$$

where τ is supposed to be a very large positive real number; which ensures $\forall x, -m_i C \leq f(x) \leq m_i C$. □

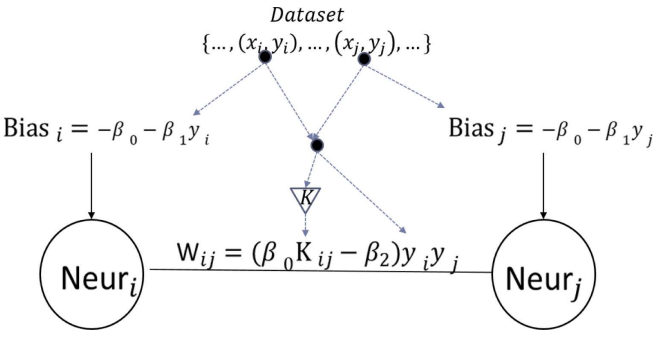


FIGURE 2. Architecture of the connection weights $W_{i,j}$ between each neurons pair.

We consider a kernel function K such that $K(x,x) = C \neq 0$.

Theorem 7. *A Continuous Hopfield network has an equilibrium points if* $W_{i,i} = 0$ *and* $W_{i,j} = W_{j,i}$.

Theorem 8. *If* $C = \dfrac{\beta_2}{\beta_0}$, *then CHN-FSVM has an equilibrium points.*

Proof. We have $W_{i,j} = (\beta_0 K_{i,j} - \beta_2) y_i y_j$ then $\forall i$ and j $W_{i,j} = W_{j,i}$ because K is symetric.

In the other hand $W_{i,i} = \beta_0 y_i K(x_i, x_i) - \beta_2 y_i^2 = \beta_0 \times \dfrac{\beta_2}{\beta_0} - \beta_2 = 0$.

Then CHN-SVM has an equilibrium point. □

3.2. Continuous Hopfield network with optimal time step

In this section, we chose, mathematically, the optimal time size in each iteration of the Euler–Cauchy method to solve the dynamical equation of the recurrent neural network proposed in this paper. At the end of the kth iteration, we know α^k and let s_k be the next step size permits to calculate α^{k+1} using the formula:

$$\alpha^{k+1} = \alpha^k + s_k \nabla E(\alpha^k)$$

s_k must be chosen such as $E(\alpha^{k+1}) \leq E(\alpha^k)$ at maximum.

As the activation function of the proposed neural network is the tanh, then: $\frac{\mathrm{d}\alpha_i}{\mathrm{d}t} = \frac{-m_i C}{\tau}(1 - \frac{\alpha_i}{m_i C})(1 + \frac{\alpha_i}{m_i C})\nabla_i E(\alpha)$.

The matrix form of the energy function is:

$$E(\alpha) = \beta_0 U^t \alpha - \frac{\beta_0}{2}(\alpha)^t T \alpha + \beta_1 y^t \alpha + \frac{\beta_2}{2}(y^t \alpha)^2$$

where $U = (1, \ldots, 1)^t \in IR^{N'}$, $\alpha = (1, \ldots, 1)^t \in IR^{N'}$, and $T_{i,j} = y_i y_j K(x_i, x_j)$ for all i and j.

At the kth iteration, the state α^k is known, then α^{k+1} is calculated by:

$$\alpha^{k+1} = \alpha^k + s_k \frac{\mathrm{d}\alpha^k}{\mathrm{d}t}$$

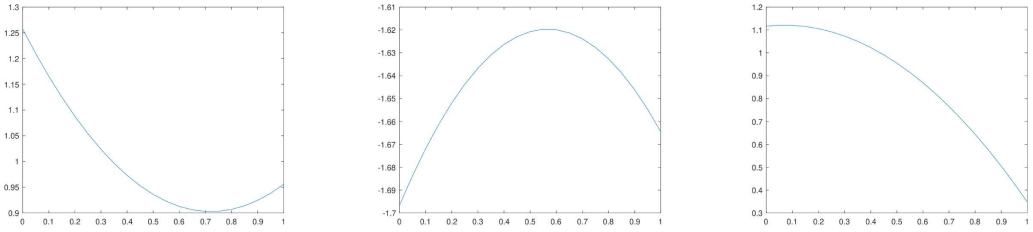
where s_k is the actual time step that must be optimal. To this end, we substitute α^{k+1} by $\alpha^k + s_k \frac{\mathrm{d}\alpha^k}{\mathrm{d}t}$ in $E(\alpha^{k+1})$:

$$e(s_k) = E(\alpha^{k+1}) = 0.5 A_k s_k^2 + B_k s_k + C_k$$

where

$$\begin{aligned}
A_k &= \beta_2 y^t \frac{\mathrm{d}\alpha^k}{\mathrm{d}t} - \beta_0 \left(\frac{\mathrm{d}\alpha^k}{\mathrm{d}t}\right)^t T \frac{\mathrm{d}\alpha^k}{\mathrm{d}t}, \\
B_k &= \beta_0 U^t \frac{\mathrm{d}\alpha^k}{\mathrm{d}t} - \beta_0 (\alpha^k)^t T \frac{\mathrm{d}\alpha^k}{\mathrm{d}t} + \beta_1 y^t \frac{\mathrm{d}\alpha^k}{\mathrm{d}t} + \beta_2 (y^t \alpha^k)\left(y^t \frac{\mathrm{d}\alpha^k}{\mathrm{d}t}\right), \\
C_k &= \beta_0 U^t \alpha^k - 0.5\beta_0 (\alpha^k)^t T \alpha^k + \beta_1 y^t \alpha^k + 0.5\beta_2 (y^t \alpha^k)^2.
\end{aligned}$$

Thus the best time step is the minimum of $e(s_k)$. The Figures 3a–3c gives different cases.



(A) (B) (C)

FIGURE 3. Graphical representation of the variation of the sums and integral terms with respect to s_k. (a) RNN-DBSVM 1. (b) RNN-DBSVM 2. (c) RNN-DBSVM 3.

3.3. Opt-RNN-DBFSVM algorithm

In this section, the procedures described in Sections 2.2, 3.1, and 3.2 are summarized into Algorithm 1.

The input of Algorithm 1 are the radius r (the size of neighborhood of current sample), the minimum of samples mp into $B(Current\_sample, r)$ (that determines the type of this sample), the three lagrangian parameters β_0, β_1, and β_2 (that makes compromise between the dual components), the bound C of FSVM [17], and the number of iterations (that represents an artificial convergence).

The Algorithm 1 processes in three macro-steps: Data preprocessing, RNN-SVM construction, and RNN-SVM equilibrium point estimation. The input of the first phase is the initial data set with labeled samples. Based on r and mp the algorithm determines the types of different samples based on the number of current sample neighbourhood discrete size. The output of this phase is a reduced sub-data set (The initial data set minus the core samples). The Input of the second phase are the reduced data set, the lagrangian parameters β_0, β_1, β_2, and the SVM bound C. Based on the energy function built in Section 3.1 and on the first and second derivatives, the architecture of CHN-FSVM is built; the bias and connection weights, which represent the output of this phase, are calculated. These later represent the input of the third phase and the Euler–Cauchy algorithm is used to calculate the degree of membership of different samples to the set of support vectors set; to ensure optimal decreasing of the energy function, at each iteration, an optimal step is determined by solving a quadratic 1-dimension optimization problem; see Section 3.2. At the convergence, the proposed algorithm produces the support vectors based on which Opt-RNN-DBFSVM can predict the class of unseen samples.

Proposition 9. *If N, r, and* ITER *represent, respectively, the size of a labeled data set* BD*, the number of the remained samples (output of the preprocessing phase, and the number of iteration, then the complexity of the Algorithm 1 is $O(r^2 * \text{ITER})$.*

Proof. First, in the preprocessing phase, we calculate, for each samples (x^i, x^j), the distance $d(x^i, x^j)$ and we execute N comparisons to determine the type of each sample; thus the complexity of this phase is $O(N^2)$.

Second, during ITER iterations, we update the activation of each neuron using the activation of all the other neurons to solve the reduced FSVM-dual, thus the third phase has a complexity of $O(r^2 \times \text{ITER})$.

Finaly, the complexity of the algorithm 1 is of $O(N^2) + O(r^2 \times \text{ITER})$. Lets denote Const-RNN-FSVM the FSVM version that implements recurrent neural network based on constant time step. Following the same reasoning, he complexity of Const-RNN-FSVM is of $O(N^2 \times \text{ITER})$. □

Notes.

– As Kernel-Adatron Algorithm (KA) is the kernel version of SMO and ISDA and KA implements two embed N-loops in each iteration, then the complexity of SMO and ISDA is of [19]. In addition, we consider L1QP-FSVM that implements numerical linear algebra Gauss–Seidel method [28], which implements two embed N-loops in each iteration, then the complexity of SMO and ISDA is of $O(N^2 \times \text{ITER})$.

Algorithm 1. Opt-RNN-DBFSVM.

Require: $mp, r, \beta_0, \beta_1, \beta_2, C, \text{ITER}$
Ensure: *Optimal support vectors*
% Density based preprocessing:
$\text{CP} \leftarrow \emptyset$; $\text{BP} \leftarrow \emptyset$; $\text{NP} \leftarrow \emptyset$;
for all s in DS **do**
if $|B(s,r) \cap DS| > mp$ **then**
$\text{CP} \leftarrow \text{CP} \cup \{s\}$
else if $mp > |B(s,r) \cap DS| > \frac{mp}{2}$ **then**
$\text{BP} \leftarrow \text{BP} \cup \{s\}$
else
$\text{NP} \leftarrow \text{NP} \cup \{s\}$
end if
end for
$RDS \leftarrow DS \setminus \{\text{NP} \cup \text{CP}\}$
% RNN-Building:
$I \leftarrow \beta_0 + \beta_1 Y$
$W \leftarrow \beta_0 K - \beta_2 Y'^2$
$m_1, \ldots, m_{RDS}$
% Optimal Euler–Cauchy to RNN stability:
$step_0 = rand(0,1)$
$\alpha'_0 \leftarrow rand(0, 1, |RDS|)$
for $k = 1, \ldots, \text{ITER}$ **do**
$D_k \leftarrow \frac{d\alpha^k}{\text{d}t}$
$A_k \leftarrow \beta_2 y^t D_k - \beta_0 (D_k)^t T D_k$
$B_k \leftarrow \beta_0 U^t D_k - \beta_0 (\alpha^k)^t T D_k + \beta_1 y^t D_k + \beta_2 (y^t \alpha^k)(y^t D_k)$
$C_k \leftarrow \beta_0 U^t \alpha^k - 0.5\beta_0 (\alpha^k)^t T D_k + \beta_1 y^t \alpha^k + 0.5\beta_2 (y^t \alpha^k)^2$
$s_k^* \leftarrow argmin_{s \in [0,1]} e(s)$
$\alpha^{k+1} \leftarrow \alpha^k + s_k^* D_k$
end for

– For a very hight size and hight density labeled data set, we have $r \ll N$, thus
$\text{ITER}_{\text{our}} \ll \text{ITER}_{\text{CHN-FSVM}}$ and $\text{ITER}_{\text{our}} \ll \text{ITER}_{\text{L1QP-FSVM}}$
$\text{ITER}_{\text{our}} \ll \text{ITER}_{\text{SMO-FSVM}}$ and $\text{ITER}_{\text{our}} \ll \text{ITER}_{\text{ISDA-FSVM}}$.
Thus complexity(our) $\prec$ complexity(CHN-FSVM) and complexity(our) $\prec$ complexity(L1QP-FSVM)
complexity(our) $\prec$ complexity(SMO-FSVM) and complexity(our) $\prec$ complexity(ISDA-FSVM).

4. Experimentation

In this section, we compare Opt-RNN-DBFSVM to several classifiers: Const-RNN-SVM (SVM based on recurrent neural network using constant Euler–Cauchy time step), SMO-FSVM, ISDA-FSVM, L1QP-FSVM, and some non-kernel classifiers (for example MLP, NB, KNN, Decision Tree. . .). The classifiers were tested on several data sets: iris, abalone, wine, ecoli, balance, liver, spect, seed, and PIMA (collected from the University of California at Irvine (UCI) [11]). The performance measures, used in this study, are accuracy, F1 score, precision, and recall.

4.1. Opt-RNN-DBFSVM *vs.* Const-CHN-FSVM

In this section, we compare Opt-RNN-DBFSVM to Const-RNN-FSVM by considering different values of the Euler–Cauchy time step $s \in \{.1, .2, \ldots, .8\}$. Tables 1 and 2 give different values of accuracy, F1-score, precision, and recall on the considered data sets. The results show the superiority of Opt-RNN-DBSVM over

TABLE 1. Performance of Const-CHN-FSVM on different data sets for different values of time step.

	SVM-FCHN $s = .1$				SVM-FCHN $s = .2$			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	96	97.66	92.9	92.00	96.8	96.03	92.20	92.00
Abalone	81.50	42.40	83.00	27.65	81.13	42.17	82.11	27.65
Wine	79.90	78.97	74.00	74.97	80.40	79.11	73.80	74.97
Ecoli	89.02	96.83	98.00	97.33	89.01	97.90	98.03	97.99
Balance	79.93	71.17	56.10	62.70	80.20	71.13	56.20	62.70
Liver	81.20	78.07	78.00	70.08	81.00	77.90	78.10	70.08
Spect	92.80	91.10	91.91	90.00	97.89	99.64	97.83	1.00
Seed	86.01	83.87	93.00	75.04	85.96	83.86	93.00	75.04
PIMA	79.95	62.00	84.93	49.6	79.86	62.10	84.00	49.6
	FSVM-CHN $s = .3$				FSVM-CHN $s = .4$			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	94.91	95.92	89.89	98.23	96.00	93.98	89.89	95.32
Abalone	78.00	51.96	83.99	30.88	82.00	41.97	83.89	33.33
Wine	80.93	77.87	74.96	74.97	81.85	77.79	73.81	74.43
Ecoli	88.90	95.81	96.97	97.33	86.93	97.87	97.95	97.95
Balance	79.98	70.99	56.00	62.32	79.88	70.87	56.83	66.23
Liver	80.97	78.83	77.79	70.56	80.78	77.94	77.98	70.08
Spect	97.92	98.99	97.96	98.56	96.78	98.78	96.89	97.79
Seed	85.79	83.68	92.84	75.88	85.96	83.79	92.87	75.61
PIMA	79.78	61.99	84.87	49.86	79.69	61.98	85.02	49.89
	FSVM-CHN $s = .5$				FSVM-CHN $s = .6$			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	94.87	95.89	89.79	98.23	95.99	93.91	89.75	95.32
Abalone	78.86	51.93	83.9	40.45	82.47	42.75	83.98	38.26
Wine	80.89	78.86	74.97	75.20	81.87	77.96	73.86	74.47
Ecoli	88.98	95.87	96.97	97.43	86.89	97.85	97.88	97.95
Balance	79.91	71.87	56.81	62.72	79.91	70.83	56.73	66.44
Liver	80.84	78.73	77.97	70.56	80.72	77.97	77.97	70.08
Spect	91.76	92.68	91.83	84.33	91.76	92.79	91.91	84.33
Seed	84.97	82.98	92.75	74.18	84.76	83.90	92.78	75.48
PIMA	79.75	61.85	84.82	49.86	79.62	61.73	84.98	48.55

Const-CHN-FSVM ($step \in STEP = \{.1, .2, \ldots, .9\}$); this superiority is quantified as follow:

$$3.43\% = \max_{(s,d)\in \text{STEP}\times\text{DATA}} (\text{accuracy(Opt-RNN-DBFSVM(s))-accuracy(const-RNN-FSVM)})$$

$$2.31\% = \max_{(s,d)\in \text{STEP}\times\text{DATA}} (\text{F1Score(Opt-RNN-DBFSVM(s))-F1Score(const-RNN-FSVM)})$$

$$7.52\% = \max_{(s,d)\in \text{STEP}\times\text{DATA}} (\text{precision(Opt-RNN-DBFSVM(s))-precision(const-RNN-FSVM)})$$

$$6.5\% = \max_{(s,d)\in \text{STEP}\times\text{DATA}} (\text{recall(Opt-RNN-DBFSVM(s))-recall(const-RNN-FSVM)})$$

where DATA is the set of different considering data sets. This results are not strange because Opt-RNN-SVM ensures an optimal decreasing of the CHN enregy function at each step.

Figures 4, A.1a–A.1c, A.2a–A.2c, A.3a–A.3c give the series of optimal steps generated by Opt-RNN-DBFSVM during iterations for different data sets. We remark that all the optimal steps are taken from the interval [.3 .4] which furnishes an optimal domain for those used a CHN based on constant step.

TABLE 2. Performance of Const-CHN-FSVM on different data sets for different values of time step.

	FSVM-CHN $s = .7$				FSVM-CHN $s = .8$			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	94.53	95.86	89.66	98.23	95.96	93.88	89.33	95.32
Abalone	77.99	51.68	83.85	30.88	81.98	41.66	83.56	33.33
Wine	80.23	77.66	74.89	74.97	81.33	77.65	73.11	74.43
Ecoli	88.86	95.65	96.88	97.33	86.77	97.66	97.83	97.95
Balance	79.75	70.89	55.96	62.32	79.66	70.45	56.1	66.23
Liver	80.51	78.33	77.9	70.56	80.40	77.67	77.90	70.08
Spect	94.36	84.60	83.77	85.99	94.36	84.60	83.77	85.99
Seed	85.71	83.43	92.70	75.88	85.71	83.43	92.70	75.61
PIMA	79.22	61.93	84.82	49.86	79.22	61.90	84.98	49.89
	FSVM-CHN $s = .9$				CHN-DBFSVM optimal value of s			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	95.96	93.88	89.33	95.32	**97.96**	**96.19**	**95.85**	**98.5**
Abalone	81.98	41.66	83.56	33.33	**98.38**	**96.07**	**96.18**	**93.19**
Wine	81.33	77.65	73.11	74.43	**96.47**	**95.96**	**96.08**	**96.02**
Ecoli	86.77	88.46	90.77	91.19	**91.82**	**97.66**	**97.83**	**97.95**
Balance	79.66	70.45	56.1	66.23	**91.31**	**90.33**	**89.54**	**90.89**
Liver	80.40	77.67	77.90	70.08	**88.10**	**85.95**	**86.00**	**85.50**
Spect	94.36	84.60	83.77	85.99	**95.55**	**86.20**	**85.28**	**86.31**
Seed	85.71	83.43	86.18	75.61	**88.90**	**84.31**	**92.70**	**84.40**
PIMA	79.22	61.90	75.90	49.89	**79.87**	**68.04**	**84.98**	**62.05**

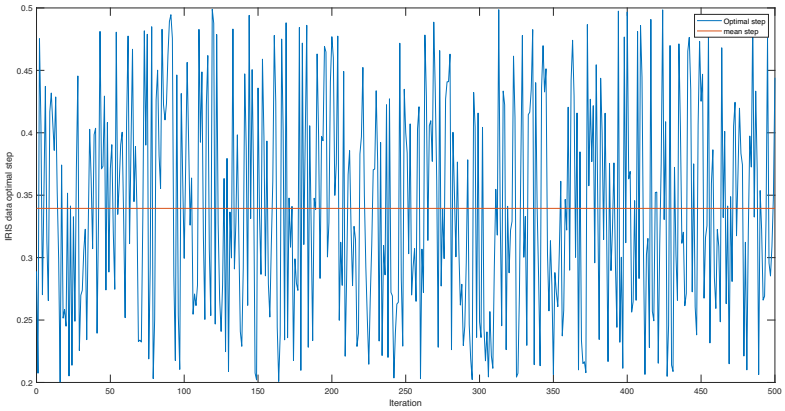


FIGURE 4. IRIS data set.

4.2. Opt-RNN-DBFSVM *vs.* Classical-Optimizer-FSVM

In this section, we give the performance of different Classical-Optimizer-FSVM (L1QP-FSVM, ISDA-FSVM, and SMO-FSVM) applied to several datasets, and compare the number of support vectors obtained by the different Classical-Optimizer-FSVM and Opti-RNN-FSVM. The Table 3 gives the values of the measures accuracy, F1 score, precision, and recall of Classical-Optimizer-FSVM on different datasets. The results show the superiority of Opt-RNN-DBFSVM.

Figures 5, 6, 7, and 8 illustrates, respectively, the support vectors obtained using L1QP-FSVM, L1QP-FSVM, SMO-FSVM, and Opti-RNN-FSVM applied to IRIS data. We note that (a) ISDA considers more than 96% as

TABLE 3. Performance of Classical-Optimizer-FSVM on different data sets.

	L1QP-FSVM				ISDA-FSVM			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	71.87	62.72	70.90	55.17	82.60	83.84	76.97	92.00
Abalone	74.75	70.90	71.88	59.30	83.80	68.72	70.60	80.66
Wine	72.97	65.89	75.63	60.11	66.78	65.90	66.61	70.02
Ecoli	66.75	55.99	61.73	41.30	51.80	48.70	33.83	51.39
Balance	65.70	53.61	60.91	41.22	50.74	58.86	68.82	60.20
Liver	64.86	52.76	60.87	40.44	50.60	48.87	62.51	51.22
Spect	70.76	62.62	67.88	50.11	77.90	71.70	75.53	70.11
Seed	70.77	58.99	67.60	45.30	80.86	81.85	79.89	79.30
PIMA	65.68	53.73	60.98	39.48	49.62	44.54	48.97	50.27
	SMO-FSVM				CHN-DBFSVM optimal value of s			
	Accuracy	F1-score	Precision	Recall	Accuracy	F1-score	Precision	Recall
Iris	71.76	62.72	70.91	55.17	**98.01**	**96.76**	**96.25**	**98.5**
Abalone	74.95	71.10	72.08	59.30	**98.78**	**96.76**	**96.98**	**93.79**
Wine	72.95	65.89	75.73	60.87	**96.77**	**96.00**	**96.88**	**96.72**
Ecoli	66.75	55.99	61.76	41.76	**92.00**	**88.85**	**90.97**	**91.19**
Balance	65.70	53.83	60.79	41.84	**91.91**	**90.93**	**89.87**	**90.97**
Liver	64.90	52.86	60.97	40.44	**88.79**	**85.98**	**86.86**	**85.58**
Spect	70.76	62.62	67.78	50.74	**95.85**	**86.60**	**85.78**	**86.51**
Seed	70.75	58.99	67.76	45.85	**88.99**	**84.89**	**86.98**	**84.87**
PIMA	65.78	53.79	60.95	39.78	**79.98**	**68.89**	**76.10**	**62.95**

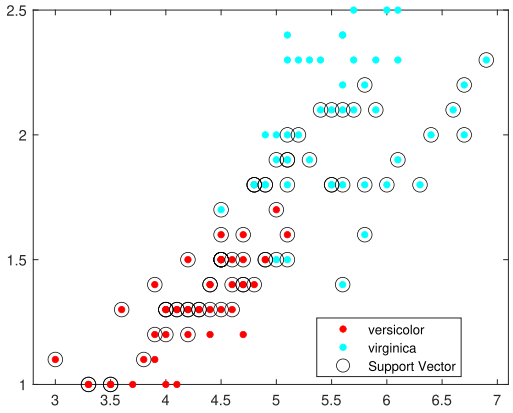


FIGURE 5. Vectors support obtained by ISDA algorithm.

support vectors, which is really an exaggeration, and (b) L1QP and SMO use a reasonable number of samples as support vectors, but most of them are duplicated, and (c) thanks to the preprocessing, Opt-RNN can reduce the number of support vectors by more than 32%, compared to SMO and L1QP, which allows it to overcome the over-learning phenomenon encountered with SMO and L1QP. Figure 8 gives the support vectors obtained by Opt-RNN-DBFSVM applied to IRIS data.

4.3. Opt-RNN-DBFSVM *vs.* Non-kernel classifiers

In this section, we compare Opt-RNN-DBFSVM to several non-kernel classifiers, namely Niave Bayes [53], MLP [47], Knn [44], AdaBoostM1 [8], Dicision Tree [7], SGDClassifier [3], Nearest Centroid Classifier [3], and

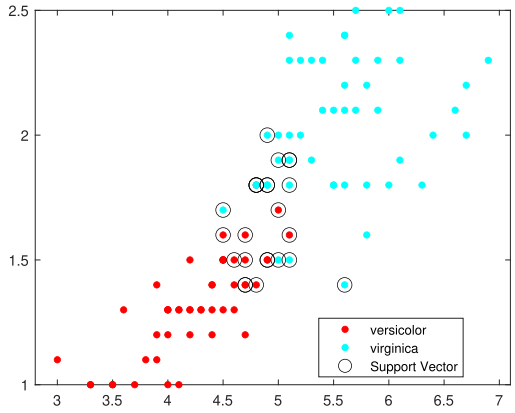


FIGURE 6. Vectors support obtained by L1QP algorithm.

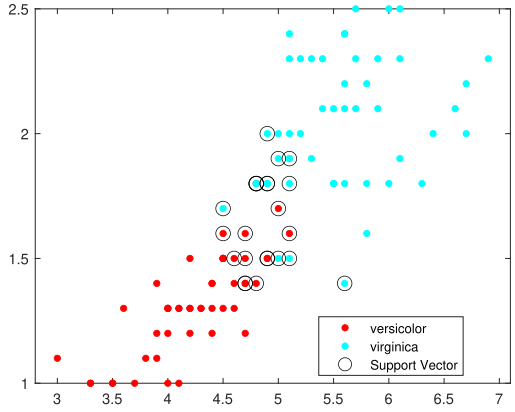


FIGURE 7. Vectors support obtained by SMO algorithm.

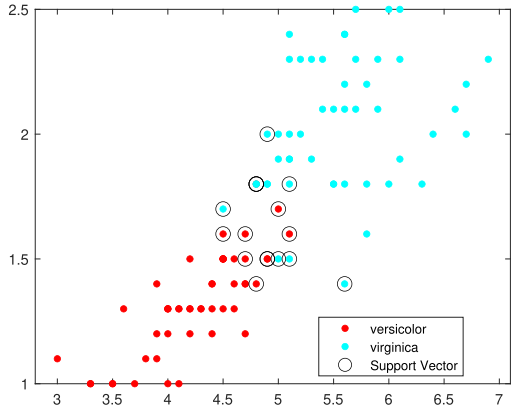


FIGURE 8. Vectors support obtained by Opt_RNN_FSVM algorithm.

Classical FSVM [48]. Tables 4–C.4 give the values of the measures accuracy, F1-score, precision, and recall for the considered data sets. The best performance is reached by Opt-RNN-DBFSVM followed by classical methods SVM, Decision Tree and Adabsot are close to Opt-RNN-DBSVM method.

TABLE 4. Comparison between Opt-RNN-DBFSVM and different classification methods on the Iris and abalone data sets.

Iris				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	90.00	87.99	77.66	1.00
MLP	26.66	0.00	0.00	0.00
Knn	96.66	95.98	91.62	1.00
AdaBoostM1	86.66	83.66	71.77	1.00
Dicision Tree	69.25	76.12	70.01	69.55
SGDClassifier	76.66	46.80	1.00	30.10
Random Forest Classifier	90.00	87.99	77.66	1.00
Nearest Centroid Classifier	96.66	95.98	91.62	1.00
Classical SVM	96.66	95.98	91.62	1.00
Opt-RNN-DBSVM	98.16	92.89	95.97	96.85
Abalone				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	68.89	51.19	41.37	67.33
MLP	62.91	47.63	36.32	47.63
Knn	81.93	53.74	70.23	43.02
AdaBoostM1	82.29	55.99	70.56	55.06
Dicision Tree	76.79	51.33	52.06	49.63
SGDClassifier	80.86	64.74	58.08	70.57
Nearest Centroid Classifier	76.07	64.79	62.60	61.15
RandomForestClassifier	82.28	57.56	71.11	48.34
Classical SVM	80.98	40.38	82.00	27.65
Opt-RNN-DBFSVM	98.88	96.86	96.97	93.91

Additional comparison study were performed on PIMA and germany diabets data sets and the ROC curves were used to calculate the AUC for the best performance obtained from each non-kernel classifier. Figures B.1 and B.3 show the comparison of the ROC curves of the classifiers DT, KNN, MLP, NB, . . . , and Opt-RNN-DBSVM method, evaluated on PIMA data set. We point out that Opt-RNN-DBFSVM quickly converges to the best results and obtains more true positives for a small number of false positives compared to several classification methods.

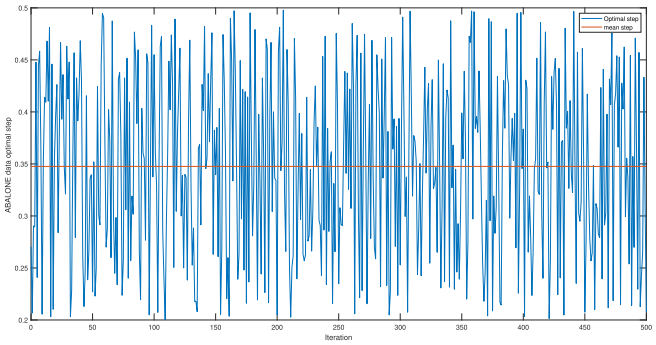
More comparisons are given in Appendix B; Figures B.2 and B.4 show the comparison of the ROC curve of the classical FSVM and Opt-RNN-DBFSVM method, evaluated on the Germany diabetes data set.

5. Conclusion

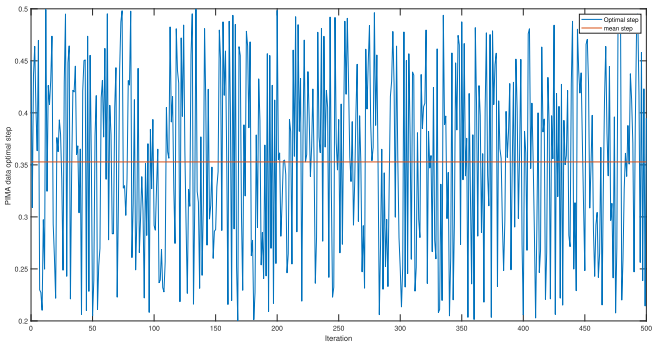
The main challenges of FSVM implementation are: the number of local minima and the amount of computer memory, required for solving the FSVM-dual, increase exponentially with respect to the size of the data set. The Kernel-Adatron family of algorithms, ISDA and SMO, has handled very large classification and regression problems. However, these methods treat noise, boundary, and kernel samples in the same way, resulting in a blind search in unpromising areas. In this paper, we have introduced a hybrid approach to deal with these drawbacks, namely Optimal Recurrent Neural Network Density Based Support Vector Machine (Opt-RNN-DBFSVM), which performs in four phases: Caracterisation of different samples, ellimination of the samples having a week probabilities to be support vector, building appropiate recurrent neural network based on original energy function, and solving the differential equation system, gouvring the RNN dynamic, using Euler–Cauchy method implimenting an optimal time step. Due to its recurrent nature, the RNN was able to memorize locations visited during previous explorations. At one hand, two main intersting fondamental results were demonstrated: the convergence of RNN-FSVM to faisible solutions and Opt-RNN-DBFSVM has a very low time complexity compared to Const-RNN-FSVM, SMO-FSVM, ISDA-FSVM, and L1QP-FSVM. At the other hand, several

experimental study were conducted based on well-known data sets (iris, abalone, wine, ecoli, balance, liver, spect, seed, pima). Based on cridible performance measures (Accuracy F1-score Precision Recal), Opt-RNN-DBFSVM out performed Const-RNN-FSVM, KAs-FSVM, some non-kernel models (cited Tab. 4). In fact, Opt-RNN-DBFSVM improved accuracy up to 3.43%, F1Score up to 2.31%, precision up to 7.52% and recall up to 6.5%. In addition, compared SMO-FSVM, ISDA-FSVM, and L1QP-FSVM, Opt-RNN-DBFSVM a reduction of the number of support vectors up to 32%, which permets to save memory for a huge applications that impliment several machine learning models. The main problem encountered in the implementation of Opt-RNN-DBFSVM is the determination of the lagrange parameters involved in the SVM energy function. In this sense, a genetic strategy will be introduced to determine these parameters considering each data set.

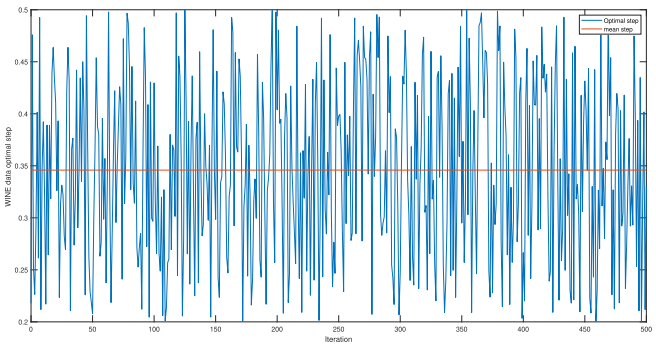
Appendix A. Optimal steps



(A)

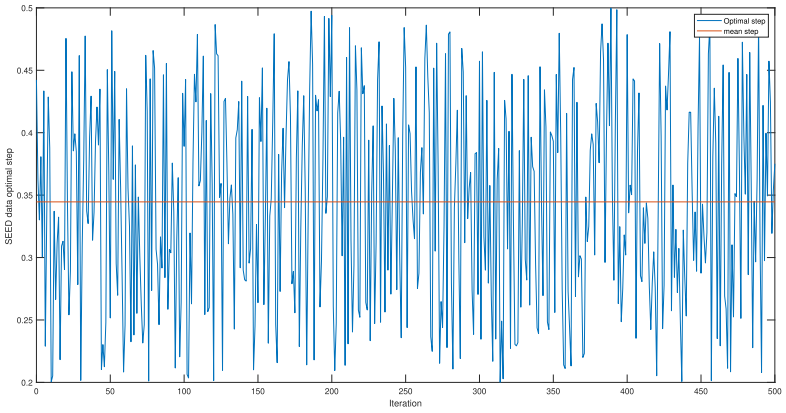


(B)

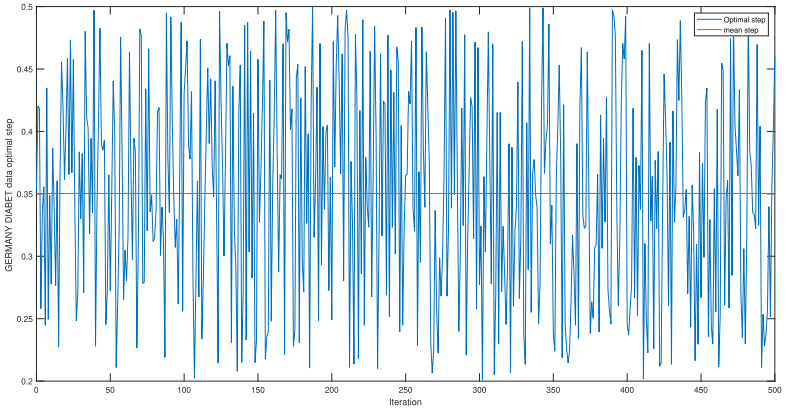


(C)

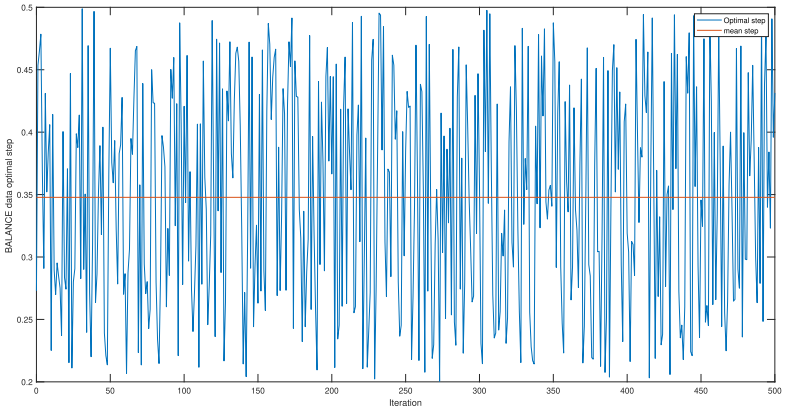
Figure A.1. (a) ABALONE data set. (b) PIMA data set. (c) WINE data set.



(A)

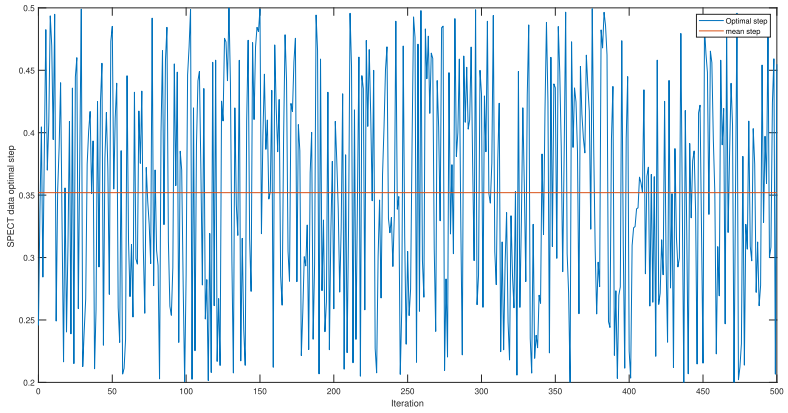


(B)

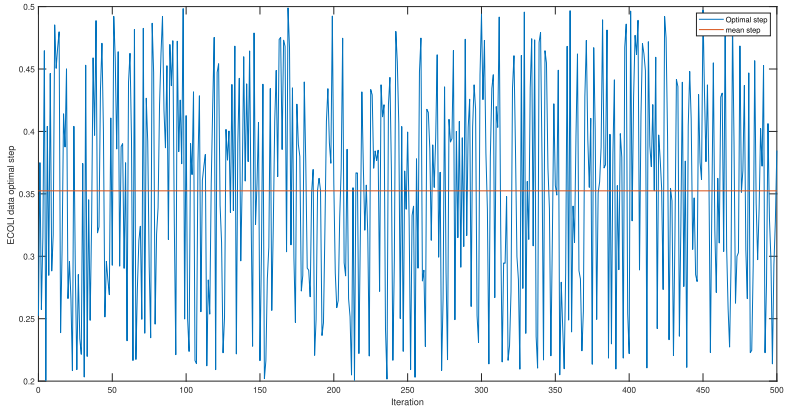


(C)

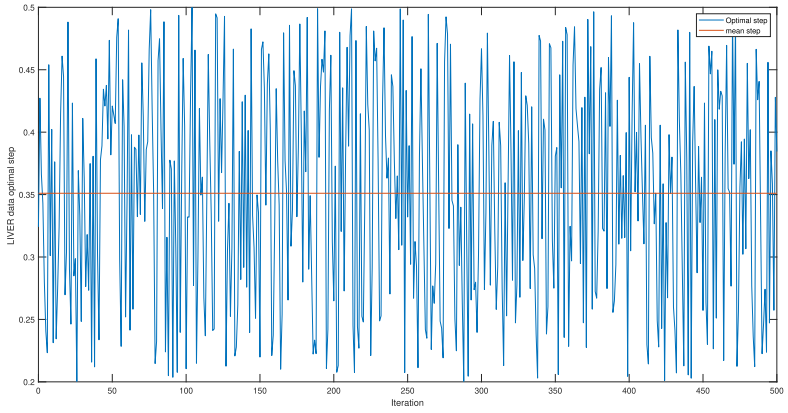
FIGURE A.2. (a) SEED data set. (b) GERMANY DIABET data set. (c) BALANCE data set.



(A)



(B)



(C)

FIGURE A.3. (a) SPECT data set. (b) ECOLI data set. (c) LIVER data set.

Appendix B. RUC curves

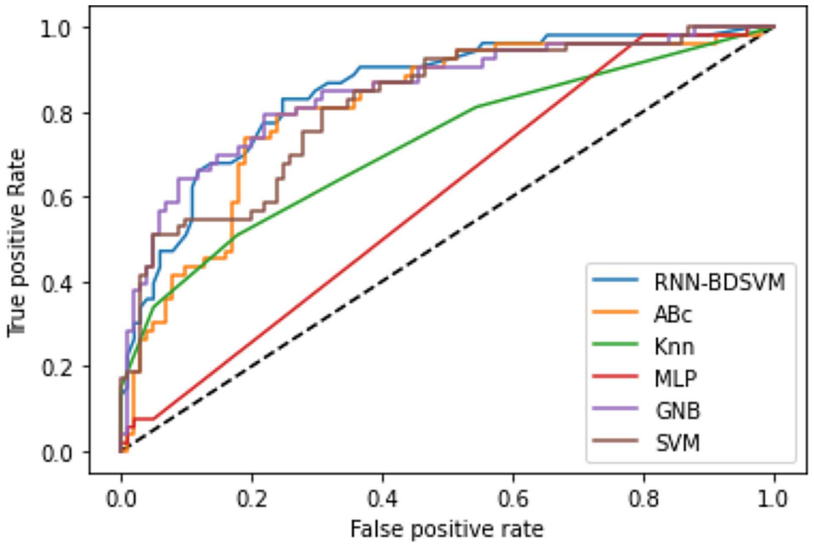


Figure B.1. RUC curve for the different classification methods applied to PIMA diabetes data set.

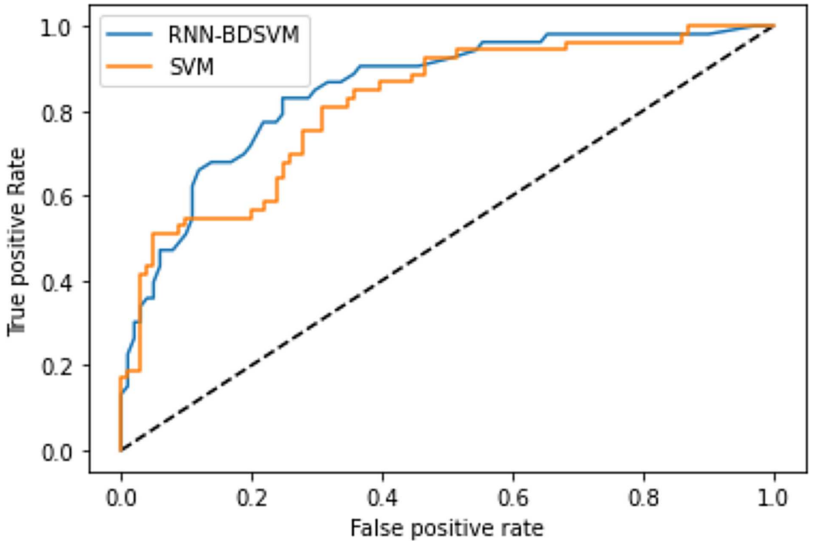


Figure B.2. Opt-RNN-DBSVM *vs.* SVM RUC curve applied to PIMA diabetes data set.

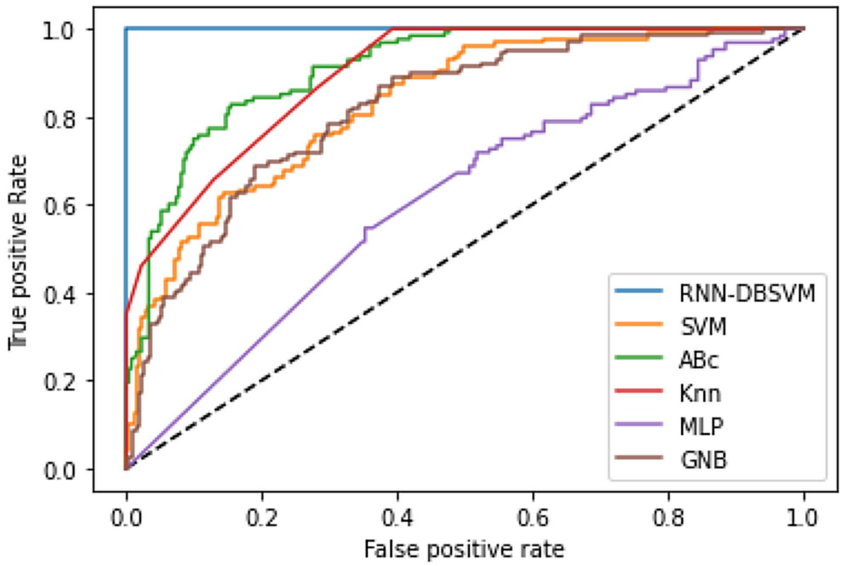


FIGURE B.3. RUC curve for the different classification methods applied to Germany diabetes data set.

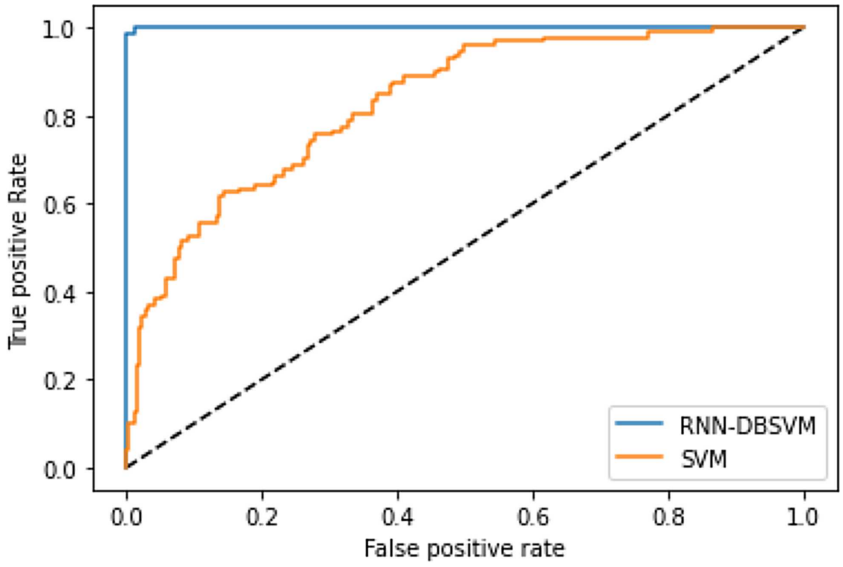


FIGURE B.4. Opt-RNN-DBSVM *vs.* SVM RUC curve applied to Germany diabetes data set.

Appendix C. Opt-RNN-DBSVM and Non-kernel Classifiers

Table C.1. Comparison between Opt-RNN-DBFSVM and different classification methods on the Wine and Ecoli data sets.

Wine				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	78.48	75.96	74.17	74.89
MLP	67.73	72.67	51.52	43.64
Knn	81.84	79.85	78.35	79.00
AdaBoostM1	88.20	70.60	81.64	76.21
Dicision Tree	83.02	81.42	79.36	80.22
SGDClassifier	68.40	83.95	52.28	44.81
Nearest Centroid Classifier	73.44	70.75	72.33	71.23
Classical SVM	79.49	78.26	73.52	74.97
Opt-RNN-DBSVM	96.47	95.96	96.08	96.02
Ecoli				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	82.08	97.11	1.00	95.66
MLP	55.22	71.45	55.00	1.00
Knn	89.55	99.87	1.00	97.02
AdaBoostM1	70.14	87.21	77.73	1.00
Dicision Tree	70.14	83.71	82.03	84.19
SGDClassifier	86.56	96.33	1.00	92.01
Random Forest Classifier	85.07	96.11	97.00	95.66
Nearest Centroid Classifier	82.08	97.28	1.00	95.39
Classical SVM	88.05	97.77	97.83	97.99
Opt-RNN-DBFSVM	91.97	88.86	90.97	91.89

Table C.2. Comparison between Opt-RNN-DBFSVM and different classification methods on the Balance and Liver data sets.

Balance				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	79.3	67.2	62.50	69.40
MLP	76.6	61.10	57.40	68.30
Knn	80.90	68.40	64.50	66.60
AdaBoostM1	81.10	69.20	70.60	66.50
Dicision Tree	79.90	64.80	72.80	68.70
SGDClassifier	69.03	65.62	66.15	65.83
Nearest Centroid Classifier	66.54	65	66.66	64.89
Classical SVM	79.70	70.7	55.60	62.70
Opt-RNN-DBFSVM	91.81	90.93	89.87	90.99
Liver				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	71.66	71.90	71.45	70.80
MLP	61.50	63.15	72.70	68.88
Knn	50.50	71.20	69.81	53.90
AdaBoostM1	88.50	89.37	89.99	79.39
Dicision Tree	39.26	45.39	48.38	48.76
SGDClassifier	49.80	60.00	49.49	50.22
Nearest Centroid Classifier	66.50	63.30	60.20	61.87
Classical SVM	80.40	77.67	77.90	70.08
Opt-RNN-DBFSVM	88.96	85.99	86.87	85.98

TABLE C.3. Comparison between Opt-RNN-DBFSVM and different classification methods on the Spect and Seed data sets.

Spect				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	63.15	77.40	96.33	65.94
MLP	97.36	99.60	97.77	1.00
Knn	92.10	96.80	97.80	95.90
AdaBoostM1	94.73	97.99	97.55	97.50
Dicision Tree	86.84	93.89	97.55	89.48
SGDClassifier	97.36	99.60	97.77	1.00
Random Forest Classifier	97.36	99.60	97.77	1.00
Nearest Centroid Classifier	57.89	72.20	1.00	72.28
Classical SVM	97.36	99.60	97.77	1.00
Opt-RNN-DBFSVM	97.86	99.97	97.97	1.00
Seed				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	85.71	79.76	92.55	69.20
MLP	30.95	0.00	0.00	0.00
Knn	85.71	79.24	73.39	85.40
AdaBoostM1	95.23	93.40	87.44	1.00
Dicision Tree	92.85	90.88	1.00	81.00
Random Forest Classifier	92.85	90.88	1.00	81.00
SGDClassifier	85.71	86.76	79.55	94.20
Nearest Centroid Classifier	85.71	79.28	92.70	75.77
Classical SVM	85.71	83.43	92.70	75.04
Opt-RNN-DBFSVM	96.99	97.79	96.87	1.00

TABLE C.4. Comparison between Opt-RNN-DBFSVM and different classification methods on the PIMA and Germany Diabet data sets.

PIMA				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	79.3	70.40	74.2	66.50
MLP	66.23	13.33	57.1	8.40
Knn	74.90	66.60	68.4	64.50
AdaBoostM1	72.72	60.37	60.8	60.80
Dicision Tree	70.77	52.63	60.2	47.60
SGDClassifier	37.66	52.47	36.98	1.00
Nearest Centroid Classifier	63.63	48.14	47.01	49.55
Classical SVM	79.22	61.90	84.7	49.6
Opt-RNN-DBFSVM	79.98	68.84	75.98	62.85
Germany Diabet Data set				
Method	Accuracy	F1-score	Precision	Recall
Niave Bayes	81.66	87.19	79.37	77.33
MLP	63.28	54.24	50.39	56.40
Knn	68.08	67.88	76.30	66.00
AdaBoostM1	91.95	90.66	92.56	90.06
Dicision Tree	55.66	53.74	59.93	50.02
SGDClassifier	79.96	79.74	70.08	78.57
Nearest Centroid Classifier	86.71	82.63	89.32	85.63
Classical SVM	78.5	59.81	74.10	50.99
Opt-RNN-DBFSVM	99.95	99.79	1.00	98.79

Acknowledgements. This work was supported by Ministry of National Education, Professional Training, Higher Education and Scientific Research and the Digital Development Agency (DDA) and CNRST of Morocco (Nos. Alkhawarizmi/2020/23).

Conflict of Interest. The authors declare that they have no conflict of interest.

References

[1] M. Aghbashlo, W. Peng, M. Tabatabaei, S.A. Kalogirou, S. Soltanian, H. Hosseinzadeh-Bandbafha, O. Mahian and S.S. Lam, Machine learning technology in biodiesel research: a review. *Prog. Ener. Comb. Sci.* **85** (2021) 100904.

[2] M. Ahmadi and M. Khashei, Generalized support vector machines (GSVMs) model for real-world time series forecasting. *Soft Comput.* **25** (2021) 14139–14154.

[3] K.M. Almustafa, Classification of epileptic seizure dataset using different machine learning algorithms. *Inf. Med. Unlocked* **21** (2020) 100444.

[4] J.K. Anlauf and M. Biehl, The AdaTron – an adaptive perceptron algorithm. *Europhys. Lett.* **10** (1989) 687–692.

[5] R. Batuwita and V. Palade, FSVM-CIL: fuzzy support vector machines for class imbalance learning. *IEEE Trans. Fuzzy Syst.* **18** (2010) 558–571.

[6] J. Bi and T. Zhang, Support vector classification with input data uncertainty. *Adv. Neur. Inf. Proc. Syst.* **17** (2005) 161–168.

[7] B. Charbuty and A. Abdulazeez, Classification based on decision tree algorithm for machine learning. *J. App. Sci. Tech. Trends* **2** (2021) 20–28.

[8] P. Chen and C. Pan, Diabetes classification model based on boosting algorithms. *BMC Bioinf.* **19** (2018) 1–9.

[9] C. Cortes and V. Vapnik, Support-vector networks. *Mach. Learning* **20** (1995) 273–297.

[10] Y. Dhanasekaran and P. Murugesan, Improved bias value and new membership function to enhance the performance of fuzzy support vector machine. *Expert Syst. App.* **208** (2022) 118003.

[11] D. Dua and C. Graff, UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2019).

[12] K. El Moutaouakil and M. Ettaouil, Reduction of the continuous Hopfield architecture. *J. Comput.* **4** (2012) 64–70.

[13] K. El Moutaouakil and A. Touhafi, A new recurrent neural network fuzzy mean square clustering method, in 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech). IEEE (2020) 1–5.

[14] K. El Moutaouakil, A. El Ouissari, A. Touhafi and N. Aharrane, An improved density based support vector machine (DBSVM), in 2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech). IEEE (2020) 1–7.

[15] K. El Moutaouakil, A. Yahyaouy, S. Chellak and H. Baizri, An optimized gradient dynamic-neuro-weighted-fuzzy clustering method: application in the nutrition field. *Int. J. Fuzzy Syst.* **24** (2022) 3731–3744.

[16] K. El Moutaouakil, M. Roudani and A. El Ouissari, Optimal Entropy Genetic Fuzzy-C-means SMOTE (OEGFCM-SMOTE). *Knowl. Based Sys.* **262** (2023) 110235.

[17] A. El Ouissari and K. El Moutaouakil, Density based fuzzy support vector machine: application to diabetes dataset. *Math. Model. Comput.* **8** (2021) 747–760.

[18] M. Ettaouil, K. El Moutaouakil and Y. Ghanou, The placement of electronic circuits problem: a neural network approach. *Math. Model. Nat. Phen.* **5** (2010) 109–115.

[19] T.-T. Frie, N. Cristianini and I.C. Campbell, The Kernel-Adatron: a fast and simple learning procedure for support vector machines, in Proceedings of the 15th International Conference on Machine Learning, edited by J. Shavlik. Morgan Kaufmann, San Francisco, CA (1998) 188–196.

[20] E.L. Glaeser, S.D. Kominers, M. Luca and N. Naik, Big data and big cities: the promises and limitations of improved measures of urban life. *Econ. Inqui.* **56** (2018) 114–137.

[21] H. Guo and W. Wang, Granular support vector machine: a review. *Artif. Intel. Rev.* **51** (2019) 19–32.

[22] K. Haddouch and K. El Moutaouakil, New starting point of the continuous hopfield network, in Big Data, Cloud and Applications: Third International Conference, BDCA 2018, Kenitra, Morocco. Springer, Cham (2018, April) 379–389.

[23] B.B. Hazarika and D. Gupta, Density-weighted support vector machines for binary class imbalance learning. *Neur. Comput. Appl.* **33** (2021) 4243–4261.

[24] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-states neurons. *Proc. Nat. Acad. Sci. USA* **81** (1984) 3088–3092.

[25] J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems. *Biol. Cybern.* **52** (1985) 1–25.

[26] T.-M. Huang and V. Kecman, Bias term b in SVMs again, in Proceedings of ESANN 2004, 12th European Symposium on Artificial Neural Networks Bruges, Belgium (2004).

[27] T. Joachims, Making large-scale SVM learning practical. Advances in kernel methods-support vector learning. http://svmlight.joachims.org/ (1999).

[28] V. Kecman, Iterative k data algorithm for solving both the least squares SVM and the system of linear equations, in SoutheastCon. IEEE (2015) 1–6.

[29] V. Kecman, M. Vogt and T.-M. Huang, On the equality of Kernel AdaTron and sequential minimal optimization in classification and regression tasks and alike algorithms for kernel machines, in Proceedings of the 11th European Symposium on Artificial Neural Networks, ESANN. Bruges, Belgium (2003) 215–222.
[30] V. Kecman, T.M. Huang and M. Vogt, Iterative single data algorithm for training kernel machines from huge data sets: theory and performance. *Support Vector Mach.: Theory App.* **177** (2005) 255–274.
[31] S. Laxmi and S.K. Gupta, Multi-category intuitionistic fuzzy twin support vector machines with an application to plant leaf recognition. *Eng. App. Artif. Intell.* **110** (2022) 104687.
[32] A.M. Law, How to build valid and credible simulation models, in 2019 Winter Simulation Conference (WSC). IEEE (2019, December) 1402–1414.
[33] Y.J. Lee and O.L. Mangasarian, SSVM: a smooth support vector machine for classification. *Comput. Opt. Appl.* **20** (2001) 5–22.
[34] C.F. Lin and S.D. Wang, Fuzzy support vector machines. *IEEE Trans. Neur. Netw.* **13** (2002) 464–471.
[35] J. Mercer, Functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. London Ser. A, Cont. Pap. Math. Phy. Char.* **209** (1909) 415–446.
[36] M. Minoux, Mathematical Programming: Theories and Algorithms. Duond (1983).
[37] E. Osuna, R. Freund and F. Girosi, An improved training algorithm for support vector machines, in Neural Networks for Signal Processing VII, Proceedings of the 1997 Signal Processing Society Workshop (1997) 276–285.
[38] J. Platt, Sequential minimal optimization: a fast algorithm for training support vector machines. Microsoft Research Technical Report MSR-TR-98-14 (1998).
[39] S. Rezvani, X. Wang and F. Pourpanah, Intuitionistic fuzzy twin support vector machines. *IEEE Trans. Fuzzy Syst.* **27** (2019) 2140–2151.
[40] S. Russell and P. Norvig, Artificial Intelligence a Modern Approach, 3rd edition. Pearson Education (2010).
[41] B. Schölkopf, A.J. Smola, R.C. Williamson and P.L. Bartlett, New support vector algorithms. *Neur. Comput.* **12** (2000) 1207–1245.
[42] B. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola and R.C. Williamson, Estimating the support of a high-dimensional distribution. *Neur. Comput.* **13** (2001) 1443–1471.
[43] B. Schölkopf, A.J. Smola and F. Bach, Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2002).
[44] A. Shokrzade, M. Ramezani, F.A. Tab and M.A. Mohammad, A novel extreme learning machine based kNN classification method for dealing with big data. *Expert Syst. App.* **183** (2021) 115293.
[45] E.W. Steyerberg, Clinical Prediction Models. Springer International Publishing, Cham (2019) 309–328.
[46] M. Tanveer, T. Rajani, R. Rastogi, Y.H. Shao and M.A. Ganaie, Comprehensive review on twin support vector machines. *Ann. Oper. Res.* (2022) 1–46. DOI: 10.1007/s10479-022-04575-w.
[47] I.O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner and A. Dosovitskiy, Mlp-mixer: an all-mlp architecture for vision. *Adv. Neur. Inf. Proc. Sys.* **34** (2021) 24261–24272.
[48] V. Vapnik, The Nature of Statistical Learning Theory. Springer Science and Business Media (1999).
[49] R.N. Verma, R. Deo, R. Srivastava, N. Subbarao and G.P. Singh, A new fuzzy support vector machine with pinball loss. *Discover Artif. Intell.* **3** (2023) 14.
[50] K. Veropoulos, *Machine learning approaches to medical decision making*. Ph.D. thesis, The University of Bristol, Bristol, UK (2001).
[51] M. Vogt, SMO algorithms for support vector machines without bias. Institute Report, Institute of Automatic Control, TU Darmstadt, Darmstadt, Germany. Available at http://www.iat.tu-darmstadt.de/vogt (2002).
[52] Y. Wang, S. Wang and K.K. Lai, A new fuzzy support vector machine to evaluate credit risk. *IEEE Trans. Fuzzy Syst.* **13** (2005) 820–831.
[53] I. Wickramasinghe and H. Kalutarage, Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation. *Soft Comput.* **25** (2021) 2277–2293.
[54] X. Xie and Y. Xiong, Generalized multi-view learning based on generalized eigenvalues proximal support vector machines. *Exp. Syst. Appl.* **194** (2022) 116491.