A Super Fast Vector Support Classifier Using Novelty Detection and No Synaptic Tuning

Radu Dogaru[1,*], Ioana Dogaru[1]
[1]*University "Politehnica" of Bucharest, Natural Computing Laboratory, Applied Electronics and Information Engineering, Bucharest, Romania*
[*]Corresponding author (E-mail : radu_d@ieee.org)

***Abstract*—A novel classifier architecture is introduced and its performances are evaluated against state of the art shallow classifiers. Its main advantage consists in a very fast learning ensured by a novelty detection algorithm, selecting a list of prototypes among the training samples, used as centers in a radial basis functions neurons layer. Only the radius of the basis functions is optimized to improve generalization in conjunction with an overlapping parameter and there is no need for synaptic tuning. Compared to state of the art models such as SVM (Support Vector Machine) or ELM (Extreme Learning Machine) our SFSVC (Super Fast Vector Support Classifier) it offers equal performance while having a more compact and fast algorithm. Thus SFSVC is well suited for embedded processing of big collections of data such as data from satellite remote sensing units, automotive sensors etc. with a good potential of being directly integrated into sensing platforms.**

***Keywords*- machine learning; neural networks; radial basis functions; extreme learning machines; support vector machines.**

I. INTRODUCTION

In various fields big amount of data has to be classified based on some labeled collections (training samples). Particularly interesting applications are in the field of image processing where large collections of pixels have to be classified into a finite set of M classes. Classification of remote sensing satellite imagery or data received from automotive sensing units, are yet another example of computationally intensive image processing algorithms. Recent works [1] investigate both the possibility to improve performance (e.g. recognition accuracy) while providing efficient and fast algorithms to cope with the high dimensionality of the available data (large sizes n of the feature vectors and large numbers N of training samples). It is therefore important to design classifiers that are capable to offer best performance (here we consider the overall accuracy – as defined in [1]) while having also a simple algorithm description (allowing thus very high speed implementations in technologies such as FPGA or GPU) and fast learning.

While recently deep learning [2] solutions are widely considered in Big Data problems, due to their ability to learn automatically the best feature extractor in the first hidden layers, they still remain computationally intensive in the learning phase. Shallow networks usually offer faster alternatives assuming that good feature extractors are designed using problem specific hand-crafted methods, adaptive projection solutions (PCA, kernel-PCA etc.) or more recently, various forms of receptive fields [3]. Consequently in this work we focus on a shallow-type classifier which can be regarded as a special case of single layer feed-forward neural network (SLFN). We show that in a C++ implementation it is faster than similar implementations of ELM or SVM and has a series of advantages making it a very good candidate for specialized hardware/software solutions (FPGA, GPU) including low-power integration with sensors.

Among shallow neural networks, extreme learning machines [4] are widely regarded as being very fast classifiers while they are governed by a very simple idea, namely replacing any complicated algorithm for tuning weights in the hidden layers with simply generating them randomly. The output layer is the only tunable, using pseudo-inverse methods. The NoProp network proposed recently [5] replaces pseudo-inverse methods with the classic LMS algorithm in the output Adaline layer while the units in the hidden layer are ELM-style trained. In a correspondence with the ELM authors, Widrow claims superiority of LMS, particularly for large number of hidden neurons. His claim is verified by us, as detailed in the ELM experiments presented herein. Other limitations of the ELM are: (i) the need to store a huge amount of randomly generated weight parameters; (ii) the need to implement the sophisticated pseudo-inverse algorithm – not very amenable for VLSI-oriented or sensor-integrated solutions. The (i) limitation also stands for NoProp and is removed in support vector approaches (parameters are readily available in the training set). Another widely used classifier is the SVM (support vector machine) which essentially constructs a hidden layer based on kernels (equivalent to RBF neurons) centered on the "support vectors". These vectors represent a sub-set from the training set selected during training such that generalization performance is maximized. Although very accurate, the SVM training algorithms have a limitation: (iii) they are complex and not very amenable for VLSI or other hardware-platform implementation. Besides, allowable kernels (not necessarily the ones optimizing the hardware implementations) are restricted by the Mercer condition to several types only. The SVM is a data oriented paradigm, so it removes the ELMs limitation (i) since the support vectors are already available in the training set. Another limitation of both SVM and ELM is: (iv) the need to optimize the regularization parameter C. In all RBF-unit based models one needs to optimize the kernel parameter γ (equivalent to RBF radius). In [6] under the name RBF-M we first introduced, another approach, recently called "fast support vector classifier"

(FSVC) and proved in a series of papers (e.g. [7][8]) to have equivalent performance to SVM and ELM while being also defined by simple learning algorithms and kernel units which are rather convenient in high-speed solutions to be implemented in hardware-oriented platforms[1]. As in SVM, FSVC selects a subset of *m* input samples as "support vectors" from a set of *N* training samples; these support vectors becoming centers of the RBF kernels. Unlike in SVM, a simple novelty detection algorithm allowing any type of kernel (RBF-unit) is employed to select support vectors thus removing limitation (iii). During a single epoch, for each new input sample the activity of the actual hidden layer is evaluated and if it is below of a certain overlapping coefficient *ov* a new support vector is added (it is actually the input sample producing this effect) corresponding to the addition of a new RBF unit on the hidden layer. Like in the NoProp, the output Adaline is tuned using LMS thus removing limitations (ii) and (iv).

Herein, SFSVC (super fast support vector classifier) was developed as a faster improvement of the FSVC classifier algorithm in [7]. Algorithms were written in C++ then embedded in .MEX files for an easy interface with Octave or Matlab. This allows a fair comparison with ELM[2] and SVM readily available implementations [9]. The SFSVC differs from FSVC in two respects: (i) a *supervised version of the novelty algorithm* for selection of centers is considered, while FSVC used an unsupervised algorithm. This approach improved both the performance and the speed of the hidden layer construction phase; (ii) There is *no tuning of the output layer*; the synapses of the output Adaline were simply initialized with the desired output values from the training set. This leads to an extremely simple learning algorithm thus eliminating most of the disadvantages of the ELM, NoProp and SVM models. The architecture and learning algorithm for SFSVC is detailed in Section II while Section III presents a synthesis of the performances, including comparisons with ELM and SVM implementations. Concluding remarks are given in Section IV.

II. SFSVC ARCHITECTURE AND TRAINING ALGORITHM

A. The classifier deffinition and architecture

In order to define the architecture and the algorithms for training and prediction the following shall be defined.

The training set $TR = \{(\mathbf{x}_k, \mathbf{d}_k)\}$ where $k = 1,..N$ is the sample index, N is the number of samples, $\mathbf{x}_k = (x_{1,k},..x_{i,k},..x_{n,k})$ is an input (feature) vector, n is the dimension of the feature vector. For classification problems it is assumed that the desired output vector is formed of 0 valued elements except $d_{j,k} = 1$ indicating that the sample k belongs to class "j" among all possible M classes. In a SFSVC structure $d_{j,k} \in \{0,1\}$ are also the synapses of the output layer thus making the product operation un-necessary (the weighted sum now becomes a sum of those terms with non-zero $d_{j,k}$ value). The test set *TS* has a similar structure yet contains different *Ns* samples and it is used only to evaluate the generalization performance of the classifier. Overall accuracy (Acc.) is considered herein as a performance measure. It represents the fraction of correctly assigned samples in the test set.

RBF units and radius: It is assumed that each hidden neuron unit is defined by a RBF function where a distance between the actual input sample $\mathbf{u} = (u_1,..u_i,..u_n)$ and the corresponding support vector (centroid) $\mathbf{c} = (c_1,..c_i,..c_n)$ can be computed in any desired way. In this work we discuss of dtype=1 distance in the case of Hamming distance $d = \sum_{i=1}^{n} |u_i - c_i|$ and dtype=2 in the case of the Euclidean one. Any other distance formula can be considered. As for the RBF functions, there is a wide palette of possibilities and no restriction, herein we consider only **rbftype=1** for a simple (hardware-oriented) triangular function defined as: $RBF(d,r) = \max(0, 1 - 0.4d/r)$ and **rbftype=2** for the classic (but not so convenient for hardware-oriented) Gaussian kernel $RBF(d,r) = \exp(-d^2/2r^2)$. In all case the radius *r* is an important parameter and is basically the main one tunable parameter that has to be optimized to ensure best generalization performance. As seen later, a finer tuning may be considered in using the *ov* (overlap factor) which is implicitly taken as $ov = 1$.

The index table TIX: To operate in prediction mode in addition to the *TR* set assumed as stored in a memory, an index table $TIX = \{i_1, i_2,..i_p,..i_m\}$ is needed (resulted after the training process); it stores integer values locating the selected support vector among the feature vectors in the *TR*. Consequently, the *p* RBF-neuron of the hidden layer has the support vector $\mathbf{c}_p = \mathbf{x}_{i_p}$ as centroid.

SFSVC Prediction algorithm:

```
1. FOR $j=1,..M$ $sc(j)=0$;  END // initialize output scores
2. FOR $p=1 .. m$
3.   $k = TIX(p) = i_p$   // locate the center in TR
4.   $d = dist(\mathbf{u}, \mathbf{x}_k)$
5.   $z = RBF(d,r)$   // calculate the output of the hidden layer
6.   FOR $j=1,..M$   // calculate the output scores
7.       IF $d_{j,k} \neq 0$ $sc(j) = sc(j)+z$ ;  END
8.   END
9. END
10. Predicted_class= Arg(max(sc))
```

Only simple arithmetic operations are invoked in the prediction algorithm, and the computation of the output scores reduces to only *m* summations. Lines 3, 4 and 7 require memory access (where the train set and the TIX matrix are stored) while the rest of the algorithm can be simply implemented as a state machine controlling the memory access. This makes the algorithm particularly suitable for FPGA implementation. In the above, lines 4 and 5 take most of the computing time so it is reasonable to consider that the

[1] A Matlab implementation of the FSVC is given at http://www.mathworks.com/matlabcentral/fileexchange/49695-fast-support-vector-classifier--a-low-complexityalternative-to-svm-/

[2] Adapted code from http://dovgalecs.com/blog/extreme-learning-machine-matlab-mex-implementation/ was considered in our implementations.

computational complexity t_2 of the algorithm is roughly $O(mn)$ i.e. linear in the number of hidden units. Experimental results confirm the above allowing evaluate the efficiency of a specific implementation by computing a specific execution time t_{ex} (expressed in ns per RBF unit, input and sample). Using the same computing platform for our SFSVC and ELM implementations a characteristic value $t_{ex} = 4.5$ is achieved while the LIBSVM from [9] for the same RBF hidden layer achieves $t_{ex} = 1.95$. This indicates that our implementation may be further optimized (speed problems may be related to the use of Eigen3 library [3], not used in LIBSVM). But this also indicates that for a fair comparisons of algorithms (regardless of their particular implementation) when compared to SVM a correction factor of 0.435 must be considered for the hidden layer times t_2 measured in both SFSVC and ELM.

B. Constructing the hidden layer via novelty detection

The training of SFSVC algorithm actually reduces to the construction of the TIX matrix. Also, the radius *r* parameter (and in addition the *ov*) must be tune until the best generalization performance is obtained on the test set. A convenient tuning procedure starts with $r = 256$ (under-fitting, small *m*) and then divides the radius by 2 while looking for performance improvement (increasing the number *m* of units). When a radius value r_o is achieved indicating over-fitting (generalization performance starts worsen) one may finely explore the radius range $[r_o, 2r_o]$ for the best accuracy. Fine tuning of *ov* in the range [0.1, 2] may also produce improvements in the accuracy. Usually, about 20 trials suffice to locate the best performance. A tuned version (herein called SFSVC-T) of the algorithm is also implemented since it was found that accuracy can be slightly improved up to the value obtained with the SVM (usually the best): The output Adaline weights are initialized as in the SFSVC model but they are corrected using the simple LMS rule during 4 additional training epochs. As seen next, the un-tuned version (SFSVC) allows very fast speeds at similar performance.

Training algorithm (TIX matrix construction): In the following the supervised training algorithm (i.e. construction of the hidden layer as a index matrix TIX) is presented for a generic class "*j*". It is first applied for *j*=1. Then the same algorithm iterates for the patterns of the next class and so on adding new elements to TIX until all samples in the training set TR are exhausted.

```
1. TIX(1) = $j_1$ // index of first sample in class "j"
2. m=1  // first hidden RBF unit
3. FOR all remaining samples (i=2,..$m_j$) in class "j"
4.     ACT =0;  // initialize the activity
5.     FOR k=1,..m  // for all existent RBF units
6.         d=dist($\mathbf{x}_{TIX(k)} - \mathbf{x}_i$)
7.         ACT=ACT+RBF(d,r)
8.     END
9.     IF ACT<ov
10.        m=m+1;  // add new RBF unit
11.        TIX(m)=i;
12.    END
13. END.
```

Lines 9-12 implement the novelty detection for creating a new hidden RBF unit.

III. Experimental results

In order to asses the performances of the SFSVC algorithm and compare with other classifiers 5 representative datasets were considered. The first two datasets are from satellite imagery problems as follows: SATIMG is a classic database from [11] (*n*=36, *M=6* classes, *N*=3217 training samples, *Ns*=3218 testing samples) while IN6 is a reduced set (*n=200, M*=16, *N=2000, M=3000*) from the original Indian Pine dataset used in [1]. In both cases a class (asphalt, lake, etc.) has to be assigned to a hyper-spectral pixel vector from the remote sensing device. In the last 3 datasets input samples are images of handwritten characters. OPTD64 is from [11] and has 8x8 pixels per character (*n*=64, *M*=10, *N*=3823, *Ns*=1797), USPS is a well known database [12] available from [9] with 16x16 pixels per character (*n*=256, *M*=10, *N*=7291, *Ns*=2007) and MNIST is a reduced set from the original MNIST[4] where a small fraction of samples was randomly selected from the original dataset (*N*=500 for training and *Ns*=100 for testing). Each sample represents an image with *n*=28x28 pixels.

A. Accuracies and optimal structures

A synthetic view of the best overall accuracies obtained for the considered databases with all classifier architectures is given in Fig.1. For each case the architecture was optimized (radius *r* and overlap factor *ov,* type of radial basis and distance) for best performance.

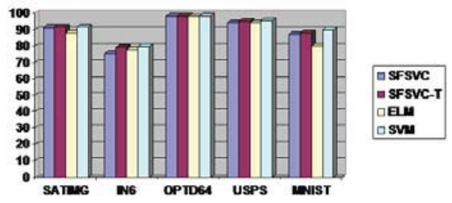

Figure 1. Overall accuracies for the optimized classifiers.

As seen, SFSVC allows obtaining overall accuracies that are only slightly under the best attainable (usually via SVM). With additional LMS tuning SFSVC-T reaches the best values. Details on the optimized structure and training times for SFSVC, SFSVC-T, ELM and SVM are given in tables I-IV. In all cases except SVM training times $t = t_1 + t_2 + t_3$ with t_1 time allocated to TIX generation (or ELM random gen. of weights), t_2 time to compute the hidden and output layer and t_3 time allocated to training of the output Adaline (or ELM training using pseudo-inverse training). Note that in the case of SFSVC very fast learning is achieved because computing the hidden layer and training is not necessary, thus $t = t_1$. For SVM, only total training time is given. Two types of RBF units are

[3] http://eigen.tuxfamily.org/index.php?title=Main_Page

[4] Available: http://yann.lecun.com/exdb/mnist/

considered: type=1 (triangular basis function + Manhattan distance) and type=2 (Gaussian basis function + Euclidean distance).

TABLE I: OPTIMAL STRUCTURES AND PERFORMANCE FOR SFSVC

Optimal	SATIMG	IN6	OPTD64	USPS	MNIST
(*r,ov,type*)	0.23, 1.2, 2	0.9, 1.3, 1	7.7, 3, 1	2.95, 0.75, 2	5.3, 0.1, 2
$t=t_1$	**0.14**	**0.14**	**0.1**	**0.56**	**0.027**
Acc. %	91.36	75.6	98.44	94.32	87
RBF units	1418	1331	1230	1141	215

TABLE II: OPTIMAL STRUCTURES AND PERFORMANCE FOR SFSVC-T

Optimal	SATIMG	IN6	OPTD64	USPS	MNIST
(*r,ov,type*)	0.25, 1, 2	1.3 4, 1	7.7, 3, 1	3.3, 1, 2	5.8, 2, 2
t_1, t_2 t_3	**0.08, 0.98, 0.62**	**0.1, 2, 0.7**	**0.06, 1.2, 0.77**	**0.38, 7.2, 1.1**	**0.02, 0.6, 0.1**
Acc. %	91.61	79.1	98.44	95.21	88
RBF units	1144	1290	1230	914	434

TABLE III: OPTIMAL STRUCTURES AND PERFORMANCE FOR ELM ($C=10^7$)

Optimal	SATIMG	IN6	OPTD64	USPS	MNIST
r (type=2)	2	5	8	18	7
T_1, t_2 t_3	**0, 0.92, 5**	**0, 2.4, 4.6**	**0, 0.9, 2.1**	**0, 7, 7**	**0, 0.54, 0.1**
Acc. %	88,16	77.9	98.1	94.22	80
RBF units	1144	1353	695	903	378

TABLE IV: OPTIMAL STRUCTURES AND PERFORMANCE FOR SVM ($C=10$)

Optimal	SATIMG	IN6	OPTD64	USPS	MNIST
γ *(type=2)*	1.62	2.8	0.046	0.01	$6.4*10^{-4}$
$T_1+t_2+t_3$	**1.29**	**3.04**	**1.45**	**8.41**	**0.86**
Acc. %	91.61	79.6	98.44	95.47	90
RBF units	1219	1346	1076	1521	356

A synthetic view of the acceleration obtained by the SFSVC algorithm with respect to the other well known algorithms (SVM, ELM) and the tuned variant SFSVC-T is given in Table V.) The same computational platform (a laptop with Pentium 2-core CPU T4300@2.1Ghz 3Gbyte RAM) was used and all algorithms were implemented in C++ compiled as .MEX files called in Octave 4 to facilitate user interface.

TABLE V: SPEED-UP OF SFSVC WITH RESPECT TO OTHER CLASSIFIERS

Speed-up (related to)	SATIMG	IN6	OPTD64	USPS	MNIST
SVM	9.21	21.7	14.5	15.0	**31.9**
ELM	42.3	**50.0**	30.0	25.0	23.7
SFSVC-T	12.0	20.0	20.3	15.5	**26.7**

As seen, our new algorithm ensures *significant speed-ups of up to 50 times* (for IN6 remote sensing problem with respect to the usually credited as very fast ELM classifier. In addition we considered a bigger version of the MNIST (25% of the original one, to fit with our memory) and the optimized SFSVC with (*r,ov,type)* = (4.55 0.15 2) trained this large dataset in only 10.6 seconds using 3559 neurons and achieving 94.2% accuracy.

IV. CONCLUSIONS

A novel type of single-layer classifier architecture, namely the SFSVC is proposed, where support vectors for RBF hidden units are selected using a fast supervised (class-dependent) novelty detection algorithm and no additional training of the output Adaline (weights in the output layer are assigned directly to the desired output values assuming that only hidden units from a class are activated for a given input sample). Results are extremely encouraging showing speedups of 10 to 50 times with respect to the ELM (usually credited as very fast learning paradigm) or SVM, while the obtained accuracies are close to best attainable using well optimized SVM (typically the loss is smaller than 1% in SFSVC) . The main speed-up reasons are: i) eliminating the computation of the hidden layer and of the output weights training; ii) the use of a supervised selection of hidden units (a speedup of t_1 around M the number of classes was determined with respect to the unsupervised version implemented in FSVC [6-8]. Another important advantage of SFSVC is the extreme simplicity of the training algorithm and the versatility of basis functions. Consequently it is very well suited for embedding in specialized computing platforms (e.g. GPU, FPGA) and for integration into satellite or automotive sensing units thus ensuring the processing of big-data directly at the sensor level, such implementations being the subject of our further research.

REFERENCES

[1] J. Lopez-Fandino, P. Quesada-Barriuso, D.B. Heras, F. Arguello, "Efficient ELM-Based Techniques for the Classification of Hyperspectral Remote Sensing Images on Commodity GPUs," in *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* , vol.8, no.6, pp.2884-2893, June 2015

[2] Y. Bengio, A. Courville, P. Vincent, "Representation Learning: A Review and New Perspectives," in *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol.35, no.8, pp.1798-1828, Aug. 2013

[3] G-B. Huang; Z. Bai, L.L.C. Kasun, C. M. Vong, "Local Receptive Fields Based Extreme Learning Machine," in *Computational Intelligence Magazine, IEEE* , vol.10, no.2, pp.18-29, May 2015

[4] Gao Huang, Guang-Bin Huang, Shiji Song, Keyou You "Trends in extreme learning machines: A review" in *Neural Networks,* vol 61, 2015.

[5] B. Widrow, A. Greenblatt, Y. Kim, D. Park, "The No-Prop algorithm: A new learning algorithm for multilayer neural networks", *Neural Networks*, Volume 37, January 2013, Pages 182-188.

[6] R. Dogaru, A.T. Murgan, S. Ortmann, M. Glesner, "A modified RBF neural network for efficient current-mode VLSI implementation", in Proceedings of the Fifth International Conference on Microelectronics for Neural Networkds and Fuzzy System (Micro-Neuro 96), IEEE Computer-Press, Lausanne 12-14 Feb. 1996, pp.265-270.

[7] R. Dogaru, I Dogaru," An efficient finite precision RBF-M neural network architecture using support vector", in *Neural Network Applications in Electrical Engineering*, Sept. 2010, pp.127-130.

[8] M. Bucurica, R. Dogaru, "A comparison between Extreme Learning Machine and Fast Support Vector Classifier", Electronics, Computers and Artificial Intelligence (ECAI), 2015 7th International Conference on, On page(s): Y-9 - Y-12.

[9] Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[10] P. Quesada-Barriuso, "Spectral-spatial classification of n-dimensional images in real-time based on segmentation and mathematical morphology on GPUs", doctoral thesis, July 2015. Available: https://citius.usc.es/sites/default/files/tesis/Tese_PabloQuesada.pdf

[11] "UCI-Machine Learning Repository", Available: http://archive.ics.uci.edu/ml/

[12] J. J. Hull. "A database for handwritten text recognition research", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550-554, May 1994.