A TRAINING PROCEDURE FOR QUANTUM RANDOM VECTOR FUNCTIONAL-LINK NETWORKS

Massimo Panella and Antonello Rosato

Department of Information Engineering, Electronics and Telecommunications
University of Rome "La Sapienza", Via Eudossiana 18, 00184 Rome, Italy
Email: {massimo.panella,antonello.rosato}@uniroma1.it

ABSTRACT

Quantum computing ideally allows designers to build much more efficient computers than the existing classical ones. By exploiting quantum parallelism and entanglement, it is possible to solve signal processing tasks on high throughput data coming from multiple sources. Random Vector Functional-Link is a neural network model usually adopted in such contexts, although quantum implementations have not been considered so far. This paper proposes a quantum version of this neural model, by introducing a specific learning algorithm to find the coefficients of the adopted quantum gates and focusing on the finite precision arithmetic imposed by the qubit strings that are used to represent the model parameters.

Index Terms— Quantum neural network, Random Vector Functional-Link, finite precision hardware, quantum learning

1. INTRODUCTION

In the general framework of quantum signal processing, developing new or modifying existing signal processing algorithms is carried out by borrowing from the principles of quantum mechanics, with the addition of some of its interesting axioms and constraints [1]. However, these novel signal processing and machine learning techniques do not inherently depend on the physics associated with quantum mechanics itself, but several advantages are obtained as in particular quantum parallelism, increased speed in learning and processing, optimized memory usage, and so forth [2].

These quantum-inspired models and methods find applications in several fields such as, for instance, frame theory, parameter estimation, covariance shaping, and multi-user wireless communication systems. Also, there are some applicative solutions in the information theory context, such as coding, error correction and cryptography [3].

As a consequence of massive parallelism, quantum neural networks ideally represent much more efficient computing systems than the existing classical ones [4, 5]. However, the implementation of quantum neural networks deals with the quantization of the model parameters, because of the digital number representation associated with quantum bits or qubits. Although some solutions have been proposed by generalizing the concept of neuron to cope with quantum principles [6], the direct quantization of coefficients on quantum architectures with finite precision arithmetic leads to poor results due to the nonlinear nature of the network [7].

Random Vector Functional-Link (RVFL) can be viewed as a feed-forward neural network with a single hidden layer, resulting in a linear combination of a fixed number of nonlinear expansions of the original input [8]. By solving the general problem of data regression, RVFLs can be applied to signal processing applications where function approximation, classification or time series forecasting is required. So far, only preliminary studies have been proposed for RVFL networks with limited hardware resources [9, 10].

In this paper, we introduce the realization of Quantum RVFL (QRVFL) networks and a methodology to address the challenging problem of model implementation with finite precision arithmetic. An optimization strategy based on a Genetic Algorithm (GA) is introduced, in order to estimate the inner parameters of the quantum gate array associated with the QRVFL network under finite precision arithmetic.

2. QUANTUM RVFL NETWORKS

RVFL systems are feed-forward neural networks where the inner layer is fixed a priori by using a predefined set of nodes. Given a d-dimensional input $\mathbf{x} = [x_1 \, \dots \, x_d]^T$, the scalar output $y \in \mathbb{R}$ is obtained by a weighted sum of C nonlinear kernels, called 'functional links', applied to the input:

$$f(\mathbf{x}) = \sum_{m=1}^{C} \beta_m h_m(\mathbf{x}; \mathbf{w}_m) \, . \tag{1}$$

A sigmoid basis function $h : \mathbb{R}^d \to \mathbb{R}$ will be adopted:

$$h(\mathbf{x}; \mathbf{w}, b) = \frac{1}{1 + \exp\{-\mathbf{w}^T \mathbf{x} + b\}} \, . \tag{2}$$

Following this formulation the RVFL model is linear in the parameters β, while the parameters $\mathbf{w}_1, ..., \mathbf{w}_C$ are randomly assigned once for all according to an uniform probability distribution. RVFL is proved to be a universal approximator if a large number C of functional links is adopted [11].

To learn an RVFL model (1) is therefore equivalent to a linear regression over the coefficients $\boldsymbol{\beta} = [\beta_1 \dots \beta_C]^T$. This is obtained by considering a training set of N input-output pairs $\{\mathbf{x}_i, y_i\}$, $i = 1 \dots N$, and solving a regularized least-squares (RLS) problem for which the optimal $\boldsymbol{\beta}$ is:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^C} \frac{1}{2} \left\| \mathbf{y} - \mathbf{H}\boldsymbol{\beta} \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta} \right\|_2^2 , \tag{3}$$

where $\lambda > 0$ is a regularization factor, $\mathbf{y} = [y_1 \dots y_N]^T$ is the vector of outputs and $\mathbf{H}$ is the so-called 'hidden matrix':

$$\mathbf{H} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_C(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \cdots & h_C(\mathbf{x}_N) \end{bmatrix} . \tag{4}$$

The learning problem (3) can be solved in a closed form, as it is strictly convex, by the well-known solution:

$$\boldsymbol{\beta}^* = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H} + \lambda \mathbf{I} \right)^{-1} \mathbf{H}^{\mathrm{T}}\mathbf{y} . \tag{5}$$

In this paper, we propose the quantum implementation QRVFL of such networks, where all the introduced parameters and input/output values adopt the digital representation of quantum computing based on strings of quantum bits (qubits). Analog-to-digital conversion and parameter quantization can be obtained by using a uniform quantizer $q_n(\cdot)$ in the respective range of any variable, with a two's complement binary representation using n qubits. It is applied element-wise if the input is either a vector or a matrix, using a same precision n for any output of the network.

The quantized output $h_m^{(n)}$, $m = 1 \dots C$, of each functional link can be rewritten as:

$$h_m^{(n)}(\mathbf{x}; \mathbf{w}, b) = q_n \left(\frac{1}{1 + \exp\left\{ -q_n(\mathbf{w}^T) q_n(\mathbf{x}) + q_n(b) \right\}} \right) , \tag{6}$$

while the generic output for an n-bit finite precision implementation of a RVFL will be:

$$f^{(n)}(\mathbf{x}) = q_n \left(\sum_{m=1}^{C} \beta_m^{(n)} h_m^{(n)}(\mathbf{x}; \mathbf{w}_m) \right) . \tag{7}$$

Considering a quantum implementation, the generic input $\mathbf{x}$ will be represented by a string $|\mathbf{x}\rangle$ of nd qubits, where each block of n qubits represents the integer corresponding to the quantized level of the jth input x_j, $j = 1 \dots d$. In the same way, the quantized output $h_m^{(n)}$ of each functional link (6) will be represented by a string of n qubits. This means that any quantum functional links will correspond to a Boolean mapping from nd-qubit to n-qubit strings.

As any Boolean function can be mapped onto a quantum gate array [12, 13], the mth functional link of the QRVFL will be represented by a unitary matrix such that $\mathbf{U}_m : |\mathbf{x}, \mathbf{0}_n\rangle \rightarrow \left|\mathbf{x}, h_m^{(n)}\right\rangle$. It is a $2^{n(d+1)} \times 2^{n(d+1)}$ square matrix for which the result of $\mathbf{U}_m$ multiplied by a $2^{n(d+1)}$ column vector $|\mathbf{x}, \mathbf{0}_n\rangle$, representing a generic quantum state in the $2^{n(d+1)}$ space of $|\mathbf{x}\rangle$ tensored with n zeros, will be a $2^{n(d+1)}$ column vector representing the quantum state of $|\mathbf{x}\rangle$ tensored with the result $\left|h_m^{(n)}(\mathbf{x})\right\rangle$. Each functional link is generated randomly, as it depends on the random generation of the hidden parameters $\mathbf{w}_1, ..., \mathbf{w}_C$ and thus, all of $\mathbf{U}_m$, $m = 1 \dots C$, can be generated randomly provided they will respect the unitary matrix constraint.

Looking at (7), the finite precision RVFL in (7) is no longer linear in the $\boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)}$ parameters, where $\mathbb{Z}^{(n)}$ represents here a generic set of integers which quantized $\boldsymbol{\beta}$ parameters can be assimilated to. In QRVFL, the quantized output $f^{(n)}(\mathbf{x})$ will be an n-qubit string $\left|f^{(n)}(\mathbf{x})\right\rangle$, which is obtained as a Boolean mapping of the functional links' outputs. Namely, a unitary $2^{n(C+1)} \times 2^{n(C+1)}$ square matrix

$$\mathbf{B} : \left|h_1^{(n)}, \dots, h_C^{(n)}, \mathbf{0}_n\right\rangle \rightarrow \left|h_1^{(n)}, \dots, h_C^{(n)}, f^{(n)}\right\rangle$$

can be defined, through which compute a $2^{n(C+1)}$ column vector containing the C outputs of functional links entangled with the QRVFL output $\left|f^{(n)}\right\rangle$.

Given the training set $\{\mathbf{x}_i, y_i\}$ and the randomly generated hidden matrices $\mathbf{U}_m$, the training of QRVFL will consist in the determination of the unitary matrix $\mathbf{B}$ that can solve the underlying Integer Least Squares (ILS) problem:

$$\begin{aligned} \min_{\boldsymbol{\beta}^{(n)} \in \mathbb{R}^C} \; & \frac{1}{2} \left\| \mathbf{y} - q_n\left(\mathbf{H}^{(n)} \boldsymbol{\beta}^{(n)} \right) \right\|_2^2 + \frac{\lambda}{2} \left\| \boldsymbol{\beta}^{(n)} \right\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^{(n)} \in \mathbb{Z}^{(n)} . \end{aligned} \tag{8}$$

ILS is a common problem in many fields of signal processing as, for example, channel coding, cryptography, radar imaging and global positioning [14, 15]. It has been shown that ILS is an NP-hard problem and the algorithms for solving it have exponential complexity [16–18]. Nevertheless, the space of solutions grows exponentially to a huge dimension, as a suited number C of functional links should be used for obtaining good approximation in spite of the relatively limited number of qubits that can be used in actual quantum hardware. Consequently, we propose an optimized training procedure, which is based on a meta-heuristic approach using GAs, in order to obtain a suited unitary matrix $\mathbf{B}$ to be adopted for quantum hardware implementation of QRVFL networks.

3. GENETIC OPTIMIZATION

GAs are adaptive search algorithms that can solve both constrained and unconstrained optimization problems by mimicking the natural selection process of biological evolution [19–21]. Any GA starts from a population of candidate solutions, called individuals, and repeatedly modifies them in an iterative process obtaining a succession of sets of individuals (i.e., the generations). The population 'evolves' over successive generations, toward an optimal solution obtained by the improvement of the fitness of the best individual.

The GA starts from a generation of completely random individuals and the successive generation is obtained through the application of selection, mutation and crossover operators. In each generation, the fitness of each individual is evaluated and a new set of solutions is created from the current one.

The previously defined Boolean mapping associates a string of n qubits with each of the 2^{nC} possible inputs coming from $\left|h_1^{(n)}, \ldots, h_C^{(n)}\right\rangle$; then, the search space has $2^{n2^{nC}}$ possible solutions. Any one of these solutions is associated with (the first) 2^{nC} columns of $\mathbf{B}$, each one having all the $2^{n(C+1)}$ entries set to 0 but one entry only set to 1. The remaining $2^{n(C+1)} - 2^{nC}$ columns are don't care and are assumed in the following to have the same structure so as to assure the unitary property. It is important to remark that, despite the matrix dimensions can be very very large, the sparsity of data allows for some compact representations and cascaded quantum architectures, which will be considered and investigated in future works.

The unitary matrix $\mathbf{B}$ we are searching for as a possible solution is an individual of the population and its entries form the so-called 'chromosome', which is a serialized vector of $2^{2n(C+1)}$ entries. More precisely, it is a sequence of $2^{n(C+1)}$ blocks, where each block has $2^{n(C+1)} - 1$ entries set to 0 and one entry only set to 1, as illustrated in Fig. 1. Also in this case, thanks to the evident sparsity, a compact and partitioned representation of data will be adopted. Moreover, each individual is admissible if and only if the corresponding matrix $\mathbf{B}$ will be meet the following constraints:

- $\mathbf{B}$ is unitary;
- for any input $\left|h_1^{(n)}, \ldots, h_C^{(n)}\right\rangle$, the output obtained by using $\mathbf{B}$ is tensored with the same input.

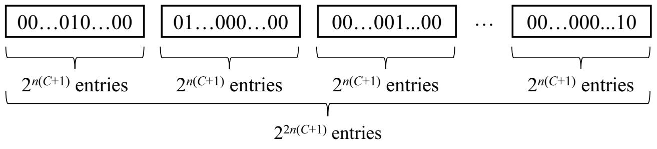

Fig. 1. Chromosome associated with a $\mathbf{B}$ solution.

Given a dataset on which the GA optimization is being performed, the randomized matrices $\mathbf{U}_m$ are determined once for all. Then, for each individual and in every generation, the quantum output is computed for all inputs of the dataset by using the particular instance of $\mathbf{B}$ associated with the chromosome of the individual; this output will correspond to a quantized output level in (7). The fitness of the individual is the Noise-to-Signal Ratio (NSR), for which the lower the NSR the better the fitness:

$$\mathrm{NSR_{dB}} = 10 \log_{10} \frac{\sum_{i=1}^{N} \left(y_i - f^{(n)}(\mathbf{x}_i)\right)^2}{\sum_{i=1}^{N} y_i^2} . \qquad (9)$$

The general steps of the GA can be summarized as:

1. A population of chromosomes G_0 with P admissible individuals is created and set as the current generation.
2. The chromosomes are evaluated by a defined fitness function and sorted by ascending values of it. A penalty is given to non-admissible individuals so that they will be certainly replaced in the future generations.
3. Some chromosomes are selected from the current generation for performing genetic operations. Cloning, mutations and crossover are applied and the produced offspring replaces their parents in the next generation.
4. The next generation becomes the current one.

The steps 2-4 are repeated for a predefined fixed number $M_{\mathbf{gen}}$ of generations. The goodness of the performance of the algorithm relies on the values of P and M_{gen} as well as on the mutation rate (M_r) and on the crossover rate (C_r). The next generation G_{k+1} is produced from the current one G_k in Step 3 of the previous algorithm as in the following:

1. The last two individuals of G_k are deleted.
2. The best individual of G_k is guaranteed to survive to the next generation, for that it is cloned and put in G_{k+1} (elitism).
3. The algorithm selects a fraction equal to M_r of the mutation rate to update the second individual of G_k by using a uniform function, then it is put in G_{k+1}.
4. A 'Roulette Wheel' procedure randomly selects a pair of parents. A probability threshold C_r is used to produce the offspring of the two parents using a 'one-point' crossover. Each of the two resulting individuals is mutated by bit swapping with a probability equal to M_r and placed in G_{k+1}. This step is repeated until the next generation contains exactly P individuals.

The successive simulations will be performed by fixing the following parameters: $P = 100$, $M_{\text{gen}} = 100$, $C_r = 0.8$, and $M_r = 0.01$, while the evolutionary process is stopped if the best fitness stalls for 5 consecutive generations within a relative interval of $\pm 0.01\%$. The optimal solution found at the end of the GA optimization will be denoted as $\mathbf{B}_{\text{gen}}$.

4. EXPERIMENTAL RESULTS

In the following, we evaluate the performances of the proposed QRVFL on four different public datasets summarized in Table 1, which are available on the UCI repository (https://archive.ics.uci.edu/ml/datasets.html).

The input and output values of datasets are normalized before training in order to accommodate every feature in the range between 0 and 1. The unitary matrices $\mathbf{U}_m$ are generated randomly and, for simplicity, their columns are an orthonormal basis of the related quantum space with either 0

Table 1. Detailed Description of Datasets

Dataset	Features	Instances	Desired output
Airfoil	6	1503	Pressure level [22]
Concrete	9	1030	Compressive strength [23]
Energy	8	768	Heating Load [24]
Istanbul	8	536	Stock exchange returns [25]

Table 2. Optimal C Found by Inner-fold Cross-validation

Dataset	n=4	n=6	n=8	n=10	float-64
Airfoil	200	200	200	300	500
Concrete	200	200	200	300	400
Energy	200	200	200	400	500
Istanbul	300	300	300	400	400

Table 3. Optimal λ Found by Inner-fold Cross-validation

Dataset	n=4	n=6	n=8	n=10	float-64
Airfoil	2^{10}	2^{9}	2^{8}	2^{-3}	2^{-10}
Concrete	2^{10}	2^{6}	2^{2}	2^{-5}	2^{-10}
Energy	2^{10}	2^{8}	2^{2}	2^{-5}	2^{-10}
Istanbul	2^{10}	2^{6}	2^{2}	2^{-2}	2^{-4}

or 1 entries. The performance is obtained through a 10-fold cross-validation, where the overall NSR is obtained comparing actual values with the predicted ones in every fold.

Taking into account the actual evolution of quantum hardware technology, four different values n of qubits are considered for the QRVFL implementation: 4, 6, 8, and 10. The performance of such solutions is compared to the one obtained by an RVFL implementation on standard computers using 64-bit floating point arithmetic, which will be assimilated to the analog result $\boldsymbol{\beta}^*$ in (5) and the corresponding output in (1).

In order to compute the optimal number of hidden nodes C and the regularization factor λ, we executed an inner-fold cross-validation of the training data. Using a grid-search procedure, we searched for the optimal C in the set $\{200, 300, 400, 500\}$ and the optimal λ as 2^{-j}, $j = -10, -9, \ldots, 9, 10$. The optimal values of C and λ for every number of qubits are reported in Table 2 and Table 3, respectively.

The results obtained by using the GA optimization, by using such choices of C and λ, are reported in Table 4. For every dataset, the NSR for values using 10-qubit quantum gate array is quite similar to the 64-bit floating point precision, with differences as low as 0.2 dB and no grater than about 4 dB for the Energy dataset. For a lower number of qubits the difference is more evident, although we are using a very strong numerical quantization within the QRVFL network.

We report in Fig. 2 the comparison of actual and predicted values of Energy dataset using a standard 64-bit floating point hardware, while in Fig. 3 there is shown the GA optimization using 8 qubits. Evidently, the behavior is comparable even with a difference of more than 5 dB.

Table 4. NSR of Genetic Optimizer Vs. Qubit Precision

Dataset	n=4	n=6	n=8	n=10	float-64
Airfoil	-25.772	-26.022	-26.394	-28.875	-30.099
Concrete	-7.879	-9.100	-11.476	-13.231	-15.350
Energy	-10.755	-13.456	-18.568	-19.802	-24.098
Istanbul	1.364	-2.225	-6.221	-6.443	-6.698

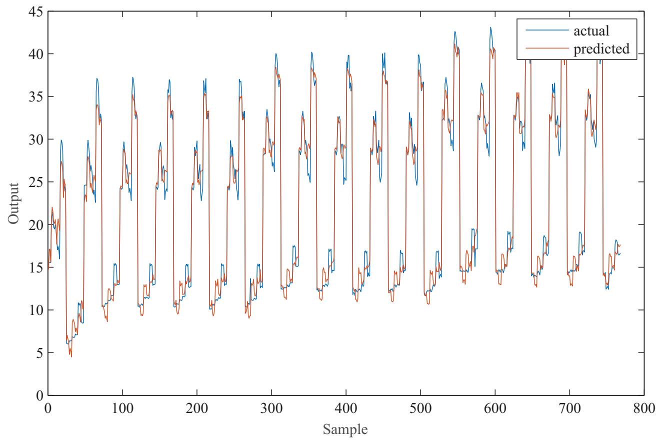

Fig. 2. Energy dataset using a 64-bit floating point RVFL.

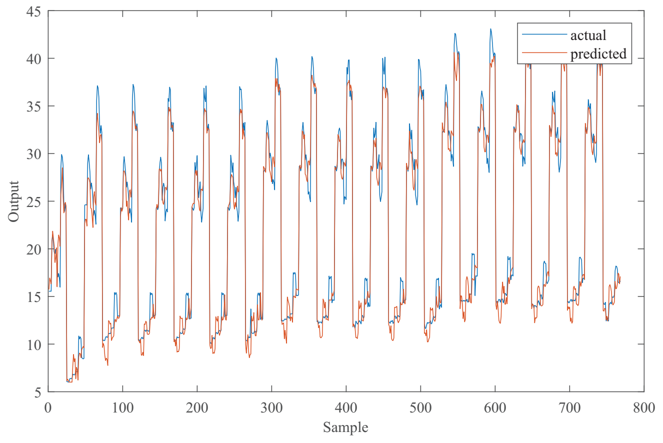

Fig. 3. Energy dataset using a 8-qubit QRVFL learned by GA.

5. CONCLUSION

In this paper, a quantum implementation of RVFL is proposed and a GA is considered to find the coefficients of the unitary quantum gates adopted in the QRVFL architecture. Experimental results show that the performances obtained by the GA optimization are close to floating point implementations even using 8 qubits only for representing the network parameters. Future works might consider unitary matrices of quantum gate arrays with complex valued entries, as for example in the Hadamard, Ising and Deutsch quantum gates, so as to consider the peculiarity of quantum rotations and other transformations of entangled qubit strings.

6. REFERENCES

[1] Y. C. Eldar and A. V. Oppenheim, "Quantum signal processing," *IEEE Signal Processing Magazine*, vol. 19, no. 6, pp. 12–32, Nov 2002.

[2] S. K. Jeswal and S. Chakraverty, "Recent developments and applications in quantum neural network: A review," *Archives of Computational Methods in Engineering*, May 2018.

[3] M.A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[4] M. Panella and G. Martinelli, "Neurofuzzy networks with nonlinear quantum learning," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 3, pp. 698–710, 2009.

[5] M. Panella and G. Martinelli, "Neural networks with quantum architecture and quantum learning," *International Journal of Circuit Theory and Applications*, vol. 39, no. 1, pp. 61–77, 2011.

[6] A.A. Ezhov and D. Ventura, *Quantum Neural Networks*, pp. 213–235, Physica-Verlag HD, Heidelberg, 2000.

[7] G.D. Magoulas, M.N. Vrahatis, T.N. Grapsa, and G.S. Androulakis, "A training method for discrete multilayer neural networks," in *Mathematics of Neural Networks*, pp. 250–254. Springer, 1997.

[8] W.F. Schmidt, M.A. Kraaijveld, and R. Duin, "Feedforward neural networks with random weights," in *Proc. of IAPR International Conference on Pattern Recognition. Conference B: Pattern Recognition Methodology and Systems*. IEEE, 1992, vol. II, pp. 1–4.

[9] J.M. Martínez-Villena, A. Rosado-Muñoz, and E. Soria-Olivas, "Hardware implementation methods in Random Vector Functional-Link Networks," *Applied Intelligence*, vol. 41, no. 1, pp. 184–195, 2014.

[10] R. Altilio, A. Rosato, and M. Panella, "A Nonuniform Quantizer for Hardware Implementation of Neural Networks," in *Proc. of European Conference on Circuit Theory and Design (ECCTD 2017)*. 2017, IEEE.

[11] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, 1995.

[12] D. Deutsch, "Quantum theory, the church-turing principle and the universal quantum computer," *Proc. of the Royal Soc. of London*, vol. A400, pp. 97–117, 1985.

[13] E. Bernstein and U. Vazirani, "Quantum complexity theory," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1411–1473, 1997.

[14] P. Teunissen, "An optimality property of the integer least-squares estimator," *Journal of Geodesy*, vol. 73, no. 11, pp. 587–593, 1999.

[15] J. Goldberger and A. Leshem, "Iterative tomographic solution of integer least squares problems with applications to mimo detection," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 8, pp. 1486–1496, Dec 2011.

[16] P. van Emde Boas, *Another NP-complete partition problem and the complexity of computing short vectors in a lattice*, Universiteit van Amsterdam. Mathematisch Instituut, 1981.

[17] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, pp. 1212–1215, Mar 2001.

[18] X.-W. Chang and Q. Han, "Solving box-constrained integer least squares problems," *IEEE Trans. Wireless Communications*, vol. 7, pp. 277–287, 2008.

[19] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press Cambridge, MA, USA, 1992.

[20] A. Rizzi, M. Buccino, M. Panella, and A. Uncini, "Genre classification of compressed audio data," in *Proceedings of IEEE Workshop on Multimedia Signal Processing (MMSP 2008)*, October 2008, pp. 654–659.

[21] R. Altilio, M. Paoloni, and M. Panella, "Selection of clinical features for pattern recognition applied to gait analysis," *Medical & Biological Engineering & Computing*, vol. 55, no. 4, pp. 685–695, Apr 2017.

[22] T.F. Brooks, D. Stuart Pope, and M.A. Marcolini, "Airfoil self-noise and prediction," 1989.

[23] I.-C. Yeh, "Modeling of strength of high-performance concrete using artificial neural networks," *Cement and Concrete Res.*, vol. 28, no. 12, pp. 1797–1808, 1998.

[24] A. Tsanas and A. Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," *Energy and Buildings*, vol. 49, pp. 560–567, 2012.

[25] O. Akbilgic, H. Bozdogan, and M.E. Balaban, "A novel hybrid rbf neural networks model as a forecaster," *Statistics and Comp.*, vol. 24, no. 3, pp. 365–375, 2014.