Neural Computing and Applications (2019) 31:9241–9260
https://doi.org/10.1007/s00521-019-04129-5

ORIGINAL ARTICLE

Evaluating generalization through interval-based neural network inversion

Stavros P. Adam[1,2] · Aristidis C. Likas[3] · Michael N. Vrahatis[2]

Received: 4 February 2018 / Accepted: 4 March 2019 / Published online: 22 March 2019

Abstract
Typically, measuring the generalization ability of a neural network relies on the well-known method of cross-validation which statistically estimates the classification error of a network architecture thus assessing its generalization ability. However, for a number of reasons, cross-validation does not constitute an efficient and unbiased estimator of generalization and cannot be used to assess generalization of neural network after training. In this paper, we introduce a new method for evaluating generalization based on a deterministic approach revealing and exploiting the network's domain of validity. This is the area of the input space containing all the points for which a class-specific network output provides values higher than a certainty threshold. The proposed approach is a set membership technique which defines the network's domain of validity by inverting its output activity on the input space. For a trained neural network, the result of this inversion is a set of hyper-boxes which constitute a reliable and ε-accurate computation of the domain of validity. Suitably defined metrics on the volume of the domain of validity provide a deterministic estimation of the generalization ability of the trained network not affected by random test set selection as with cross-validation. The effectiveness of the proposed generalization measures is demonstrated on illustrative examples using artificial and real datasets using swallow feed-forward neural networks such as Multi-layer perceptrons.

Keywords Neural networks · Generalization · Inversion · Interval analysis · Reliable computing

Abbreviations

HPD	Highest posterior density
INTLAB	INTerval LABoratory
IA	Interval analysis
MLP	Multi-layer perceptron
OTS	Off training set
PDF	Probability density function
SCS	Set computations with subpavings
SIVIA	Set inversion via interval analysis

✉ Stavros P. Adam
adamsp@uoi.gr; adamsp@upatras.gr

Aristidis C. Likas
arly@cs.uoi.gr

Michael N. Vrahatis
vrahatis@math.upatras.gr

[1] Department of Informatics and Telecommunications, University of Ioannina, Arta, Greece

[2] Computational Intelligence Laboratory, Department of Mathematics, University of Patras, Patras, Greece

[3] Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece

1 Introduction

The classification performance of some network architecture is measured by estimating the error of classification on previously unseen data [4]. This error is also identified as generalization error and defines the ability of a network to generalize. For some specific network architecture generalization depends on the efficiency of the training algorithm as well as on the degree the available training data represent the true probability distribution of the underlying problem.

Due to lack of a concise mathematical definition of generalization, which would permit a deterministic evaluation, the dominant approach for evaluating generalization is a statistical method, the well-known cross-validation. This is a hold-out technique relying on a suitable division of the dataset into two subsets that are the training set and the test set. Measuring the generalization error, successively, on several test sets (called folds) and averaging the

resulting measurements gives an estimation of the network generalization ability.

However, despite the fact that attempts have been made to prove that cross-validation results in a consistent estimator of the learning algorithm's generalization, it seems that this approach does not sufficiently define generalization. Several causes are identified by Wolpert [34] for this problem which are, mainly, related to the difference of the distribution generating test patterns during cross-validation from the distribution of the OTS defined by real-world processes.

Another reason for questioning the efficiency and unbiasedness of cross-validation is the stochastic splitting into folds, while, to a lesser extent, one may consider that cross-validation is computationally intensive since it requires the training process to be repeated several times. Finally, to the best of our knowledge, there is not any work suggesting that cross-validation can be used to assess the difference in terms of generalization between two networks that are already trained and, possibly, featuring similar classification performance.

In this paper, we look at the problem of assessing the generalization ability of neural networks aiming to show that it is possible to avoid deficiencies of cross-validation using a deterministic approach. To this end, we propose a set membership approach consisting in the definition of the area of the input space, i.e., the set of all patterns, driving the network output activity to values which are higher than some predefined threshold. This threshold defines the certainty level above which the values of some class-specific output clearly vote for this class against the others. The area of the input space satisfying these requirements is called *the domain of validity* of the neural network [1, 5]. The proposed approach derives verified results regarding the domain of validity in the sense that for each class it contains exactly the area whose patterns are guaranteed to activate the correct network output. This validity domain can be reliably computed through IA-based inversion of a network, as proposed in [1]. Having reliably computed this area one is certain that there are not any other parts in the input space affecting generalization.

More precisely, estimation of the generalization ability using the proposed method focuses on MLP networks trained to perform some pattern classification task. For such a task with M classes, we assume an MLP with M logistic outputs and 1-of-M target encoding. Thus, each output corresponds to a specific class and the network is trained (using the available dataset) to provide high values (ideally equal to 1) for patterns of the corresponding class only. Once the network training has been completed, *we aim to evaluate the generalization ability of the final network* by performing network inversion and focusing on the parts of the input space forming the *network validity domain*. MLP inversion refers to computing the inverse mapping of a given MLP, i.e., a mapping from the output domain to the input space. Note that the employed IA-based inversion technique is reliable in the sense that it provides verified results in a guaranteed way, as the interval computations permit to automatically verify the results obtained [3, 16].

A very convenient characteristic of the interval-based MLP inversion approach is that it provides a partitioning of the validity domain as a set of non-overlapping hyperboxes with varying size and density. Such a partition provides important information on what the network has learned, that we exploit to define (in conjunction with the available dataset) measures of generalization performance taking into account issues such as under-training and over-training. In order to validate the proposed measures, we carried out a number of illustrative experiments using artificial and real datasets. The experimental results provide significant evidence on the potential of the derived measures to assess the quality of trained MLP classifiers.

The approach presented in this paper constitutes a deterministic method which can be used to assess the generalization of a network after training without having to apply cross-validation on various instances of its architecture. Thus, the entire dataset can be used for training, there is no need for separate testing and so, two different network architectures can be compared in terms of generalization in a deterministic way by considering the area of the input space effectively seen by each architecture.

The paper is organized as following. Section 2 defines the problem background and related literature. In Sect. 2.1, we examine the problems that lead to the introduction of a new approach, while in Sect. 2.2 we analyze the considerations on which the proposed approach is built. The application inclined reader may omit this section. Section 3 is dedicated to the description of the basic interval arithmetic concepts and the inversion procedure based on IA along with related references. In Sect. 4, we describe and explain the proposed generalization measures. Section 5 is devoted to the experimental evaluation and the discussion of the results obtained. Finally, Sect. 6 provides conclusions and directions of future work.

2 Problem background and literature review

2.1 Generalization assessment in a classification context

For the elaboration of the proposed approach, we consider MLP neural classifiers trained on a classification problem with M classes, $C_1, C_2, \ldots, C_M$. The problem is defined as a

set $\mathbf{D} = (\mathbf{X}, \mathbf{Y})$ of P examples of N-dimensional patterns $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_P\}$, where each one is known to belong to one of the M classes. The P examples instantiate the random variable $\mathcal{X}$, and the desired outputs $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_P\}$ the random variable $\mathcal{Y}$. Moreover, it is assumed that the joint pdf $p(\mathcal{X}, \mathcal{Y})$ constitutes a faithful representation of the original distribution function of the classification problem space.

When training a neural network one expects that it will be able to correctly approximate/predict output at unseen observations of input. This property known as generalization is affected by two contradictory issues as nicely stated in the well-known bias/variance dilemma which refers to the decomposition of the expected generalization error into two components: the bias and the variance. Under-training refers to the errors produced by the inefficiency of the network to correctly approximate the decision boundaries. Usually, this is due to the lack of an appropriate network model or poor training. Over-training refers to the fact that a large network usually learns specific details of the training examples (and perhaps the noise in the dataset) and disregards the regions in the vicinity of the examples which are very likely to be correctly classified. Both under-training and over-training contribute to the generalization error and constitute the two sources of the estimation error one needs to consider when designing and training a neural network and more specifically an MLP.

Concerning generalization several studies are reported in the literature with rather interesting results. For instance, Wolpert in [31, 32] attempted to derive a mathematical theory on generalization and many of the results were used in subsequent works [33–35] where generalization and cross-validation were closely investigated regarding the ability to evaluate the performance of learning algorithms. Knowledge of the distribution of the input data and the prior of the classes of the problem constitute important prerequisites for accurately assessing generalization. Other important factors include, the confidence that the distribution of the training patterns is a good approximation of the distribution of the input space, the effectiveness of the training algorithm or even the fitness of the network architecture with the problem at hand, see [8] and references therein. Finally, one should note that the study of generalization is still an active subject of research [14, 22].

In its origin, cross-validation is a statistical technique for estimating generalization [20]. However, in practice, it is K-fold cross-validation that seems to be the standard for evaluating generalization of a neural network architecture. On the other hand, generalization performance of a learning algorithm is defined in terms of its OTS error, Wolpert [34]. Given that no actual test data exist, K-fold cross-validation divides the available dataset in K parts of equal size, which are the folds. Then, the $K-1$ folds form the training set and the K-th fold is left aside for testing. This process is repeated K times and the final test error is the average of the K test errors corresponding to the test sets extracted from the available dataset. There are several reasons for which K-fold cross-validation is disputed that it is an efficient and unbiased estimator of the generalization error.

- In real-world problems, usually the training set is not produced by the same process as the one giving the test data. So, in these situations, the use for testing of data extracted from the training set, especially when these two sets overlap, is not justified to be unbiased, Wolpert [34].
- In order to achieve statistically significant results, cross-validation needs to perform several independent data splits with test error monitoring. These data splits result in significant waste of data, inefficient training and yet a potentially biased estimation of the generalization error.
- For real-life applications, only a limited amount of training examples is available while according to the theory of stochastic approximation [17, 25] a good network design should rely on a big number of input data. In practice, this leads to techniques trying to increase the available dataset by some sort of data recycling at the expense of assigning excessive probability mass on the exact points of the training set examples. As a result, neural network design is restricted on excessively weighted examples which entails over-fitting and poor generalization ability for the resulting network [15].
- Training a neural network using gradient descent with random initial weights is not deterministic. This issue affects cross-validation error which is very likely to be different from generalization error. So, similarity between the distributions governing these two errors is a rough hypothesis.
- Finally, cross-validation is used for determining, through experiments, among several competing neural network architectures the most suitable for solving a specific problem. Hence, this statistical method is not suitable for determining among two already trained neural networks which one performs better in terms of generalization.

2.2 Considerations supporting the new approach

Typically, in problems involving MLP classifiers, an input example $\mathbf{x}$ is classified according to the maximum output rule: "$\mathbf{x}$ is assigned to class C_j if the jth component of the network output is greater than all the other components"

[9]. In addition, for output values in the interval [0, 1], the MLP outputs are considered to be equivalent to Bayesian posterior probabilities of the corresponding classes [24]. As shown in [7] by Hampshire and Pearlmutter, when the conditions: (a) perfect training, (b) sufficiently large dataset and (c) complex network, are met then the relation between the *a posteriori* class probabilities and the network output values is linear and so $P(\omega = \omega_j|\mathbf{x})$ is mapped to the interval [0, 1]. However, in practice, due to various perturbations, the actual output values are non binary and so the conditional probability $P(\omega = \omega_j|\mathbf{x})$, while still being linear, it is mapped to the interval $[\epsilon, 1-\epsilon]$ for some appropriate value of ϵ (e.g., $\epsilon = 0.2$). Note that similar conclusions were, also, provided by Richard and Lippmann in [24] regarding the degradation of the network estimation accuracy.

For the majority of the classification tasks, it is commonplace when assigning the input pattern to the jth class to consider the value of the jth output to be in the interval $[1-\beta, 1]$, defined for some suitably chosen value of β (e.g., $0 < \beta \leqslant 0.4$). This provision constitutes an intuitive, yet practical and efficient way, to deal with uncertainty of the network decision due to various reasons. On the other hand, for the same classification rule, an output node is considered to be inactive if its activation is in the interval $[0, \beta]$. Assuming that β is chosen so that $\epsilon \leqslant \beta$, for ϵ as defined in [7] then it is legitimate to argue that "this interval heuristic complies with the aforementioned theoretical results."

As a consequence, if $[1-\beta, 1]$ is the interval of valid network decisions, it is straightforward to think of the input data providing valid network decisions. The area of the input space consisting of those data for which the network provides a valid output is called *the domain of validity* of the network for this output. So, for any class, say C_j, of the classification problem and a specific network trained on this problem, there exists a domain of validity defined by the interval $[1-\beta_j, 1]$ for the jth output node. For simplicity, here we will assume that for all classes we have $\beta_1 = \beta_2 = \cdots \beta_M = \beta$. Finally, the domain of validity of the network is the union of the domains of validity of the individual classes. Note that the area of the input space which does not belong to any of the M domains of validity corresponds to unclassified input data and corresponds to output values in the interval $[\beta, 1-\beta]$. The domain of validity of the network taken as a union of disjoint domains together with the area corresponding to the unclassified input data forms a partition of the input space.

The answer to the natural question of how one can calculate the domain of validity of a neural network and more specifically of an MLP has been addressed by the subject known as *neural network inversion*. Inversion of a trained MLP has been the objective of several research efforts with the aim to show that it permits to define the area in the input space effectively covered by the network function [5, 21]. As a consequence, one should be able to sketch the decision boundaries learned by the network and to formulate rules governing the relation of the classification decision with specific values of the data features. Hence, neural network inversion has been used either to provide qualitative conclusions of the neural classification function [13, 18, 23] or to extract provably correct IF-THEN rules [6, 27, 29] which explain operation of the MLP.

The majority of research on neural network inversion provides only rough estimates of the domain of validity of an MLP. This is due to the complexity of the inversion procedure itself, as the whole input space needs to be investigated. Another reason can be attributed to the fact that a very fine grained, level of explanation of the neural network operation would result in the formulation of a very large number of IF-THEN production rules which would be inefficient to be exploited by some decision support mechanism. On the contrary, effective determination of such rules results in less detailed definition of the validity domain [10].

In [1], the use of the SIVIA approach [11] has been proposed for inverting an MLP and defining its domain of validity. Actually, this method of MLP inversion constitutes a set membership technique based on IA which, given some interval of the MLP output activity, uniquely determines a consistent and guaranteed domain in the input space. Hence, given the interval of the valid network outputs, the use of SIVIA provides a reliable estimation of the domain of validity as it permits to define the regions of validity in a guaranteed way, while at the same time it permits to obtain a quantitative estimation of the domain of validity of the neural network. As we show in the following section the interval-based inversion method approximates the domain of validity producing a set of n-dimensional hyper-boxes that can be exploited to define measures of the generalization ability of the network.

3 Interval-based inversion and SIVIA

3.1 Basic interval analysis concepts

Interval arithmetic was introduced as a means to perform numerical computations with guaranteed accuracy and bounding the ranges of the quantities, used in the computations. An interval or interval number I is a closed interval $[a, b] \subset \mathbb{R}$ of all real numbers between (and including) the endpoints a and b, with $a \leqslant b$. The arithmetic defined on intervals, rather than real numbers, is called interval

arithmetic. Whenever $a = b$ the interval is said to be degenerate, thin or even point interval. Regarding notation, an interval is usually denoted $[\underline{x}, \overline{x}]$ or $[x]$. Usually, an interval object (variable, vector, matrix, etc) is denoted by $\mathbf{x}$ to differentiate it from the explicit bracketed notation $[\underline{x}, \overline{x}]$. An interval $[\underline{x}, \overline{x}]$ where $\underline{x} = -\overline{x}$ is called a symmetric interval. An n-dimensional interval vector $\mathbf{V}$ is a vector having n components $(\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n})$ such that every component $\mathbf{v_i}, 1 \leqslant i \leqslant n$ is a real interval $[\underline{v_i}, \overline{v_i}]$. We need to note that $\mathbb{IR}^n$ denotes the set on n-dimensional vectors of real intervals. The definition of interval objects such as vectors, matrices, functions, etc. and their subsequent study resulted in the establishment of IA.

In practical calculations, interval arithmetic operations are reduced to operations between real numbers [11]. For the intervals $[\underline{x}, \overline{x}], [\underline{y}, \overline{y}]$ it can be shown that the following intervals are produced for each arithmetic operation:

$$[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{1a}$$

$$[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{1b}$$

$$[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] = \left[\min\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right), \max\left(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}\right)\right] \tag{1c}$$

$$[\underline{x}, \overline{x}] \div [\underline{y}, \overline{y}] = [\underline{x}, \overline{x}] \times \frac{1}{[\underline{y}, \overline{y}]}, \text{ with} \tag{1d}$$

$$\frac{1}{[\underline{y}, \overline{y}]} = \left[\frac{1}{\overline{y}}, \frac{1}{\underline{y}}\right], \text{ provided that } 0 \notin \left[\underline{y}, \overline{y}\right] \tag{1e}$$

If $[x] \subseteq D$ is an interval in the domain of a real function $f : D \subset \mathbb{R} \to \mathbb{R}$ then $f([x])$ is used to denote the range of values of f over $[x]$. Computing such a range, $f([x])$, using IA tools means to enclose it by an interval which is as narrow as possible. This constitutes an important matter in IA as it is used in various problems: localization and enclosure of global minimizers of f on $[x]$, verification of $f([x]) \subseteq [y]$ for given $[y]$, nonexistence of a zero of f in $[x]$ etc.

In order to enclose $f([x])$, one needs to define a suitable interval function $[f] : \mathbb{IR} \to \mathbb{IR}$ such that $\forall [x] \in \mathbb{IR},\ f([x]) \subset [f]([x])$, see Fig. 1. The interval function $[f]$, which is called an inclusion function of f, makes it possible to compute a box $[f]([x])$ which is guaranteed to contain $f([x])$, whatever the shape of $f([x])$, [11]. If $f(x), x \in D$ is computed as a finite composition of elementary arithmetic operators $\{+, -, \times, \div\}$ and standard functions such as {exp,sqr,cos,sin,..}, then the inclusion function, which is obtained by replacing in f the real variable x by an interval variable $[x] \subseteq D$ and each operator or standard function by its interval counterpart, is called a *natural inclusion function* of f. As noted in [11] the natural inclusion function has important properties such as being

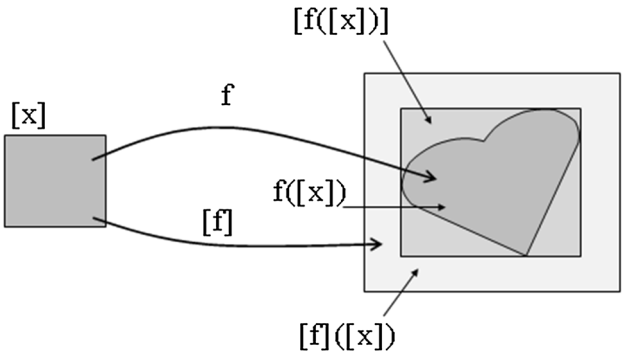


Fig. 1 A function f, an inclusion function $[f]$ and the images of $[x]$

inclusion monotonic, and if f is defined using only continuous operators and continuous standard functions the natural inclusion function is convergent.

3.2 The SIVIA approach

SIVIA is a set membership method introduced by Jaulin and Walter [12] in order to allow for the guaranteed estimation of nonlinear parameters from bounded error data. The method relies on IA and its application results in defining an axis-aligned box or a union of axis-aligned boxes approximating a set of interest. Hence, given a function $f : X \to Y$, where $X \subset \mathbb{R}^n$, $Y \subset \mathbb{R}^m$ and an interval vector, i.e., a box, $[y] \subseteq Y$, the objective is to define the set of unknown vectors $x \in X$ such that $f(x) \in [y]$. This set can be defined as $S = \{x \in X \subseteq \mathbb{R}^n | f(x) \in [y]\} = f^{-1}([y]) \cap X$, where X is the search space containing the set of interest S; $[y]$ is known in advance to enclose the image of the set $f(S)$ and S denotes the unknown set of interest. Note that, here, f^{-1} denotes the reciprocal image of f, as f may not be invertible in the classical sense.

The solution proposed by SIVIA for this problem consists in computing boxes and unions of boxes S^- and $S^+ = S^- \cup \Delta S$ which form guaranteed inner and outer approximations of S as they satisfy the relation $S^- \subseteq S \subseteq S^+$, [11]. SIVIA is a branch-and-bound approach which computes enclosures by recursively exploring the whole search space. During computation, a box $[x] \in \mathbb{R}^n$ is designated as,

- feasible if $[x] \subseteq S$ and $f([x]) \subseteq [y]$,
- infeasible if $[f]([x]) \cap [y] = \emptyset$ and,
- in all other cases, $[x]$ is said to be indeterminate which means that $[x]$ may be feasible, unfeasible or ambiguous.

The condition $[x] \subseteq S$ and $f([x]) \subseteq [y]$ is necessary and sufficient for $[x]$ to be feasible. Feasible boxes are added to S^- and infeasible become members of the complement of S^+ denoted N. Finally, any indeterminate box is bisected and the method recursively examines the two resulting sub-

boxes. Bisection is possible up to some limit, which is preset for the problem and defines its resolution. Boxes that are indeterminate and cannot be further bisected are added to the union ΔS. The method described above is more formally given by the algorithm 1.

For a more detailed description of the algorithm implementing SIVIA the reader should refer to [11]. It is worth noting here that SIVIA applies to any function f for which an inclusion function $[f]$ can be computed.

Algorithm 1 SIVIA(in: $[f]$, $[y]$, ε, $[x]$; inout: S^-, ΔS, N)

Require: Before calling SIVIA initialize $[f]$, $[y]$, ε; and set $S^- := \emptyset$; $\Delta S := \emptyset$; $N := \emptyset$;

1: **if** $[f]([x]) \subset [y]$ **then**
2: $S^- := S^- \cup [x]$;
3: return;
4: **end if**
5: **if** $[f]([x]) \cap [y] = \emptyset$ **then**
6: $N := N \cup [x]$;
7: return;
8: **end if**
9: **if** width$([x]) < \varepsilon$ **then**
10: $\Delta S := \Delta S \cup [x]$;
11: return;
12: **end if**
13: bisect $[x]$ getting left and right sub-boxes $[x]_1$ and $[x]_2$
14: SIVIA(f, $[y]$, ε, $[x]_1$, S^-, ΔS, N);
15: SIVIA(f, $[y]$, ε, $[x]_2$, S^-, ΔS, N);

An example of the application of SIVIA approximating the interior of a circle having its center at the beginning of the axes and radius equal to 1 is shown in Fig. 2. In this Figure, the red area is defined by SIVIA and corresponds to the points (x_1, x_2) in the input area $[-2, 2] \times [-2, 2]$ for which $f(x_1, x_2) \in [0, 1]$ i.e., the interior of the circle. Note

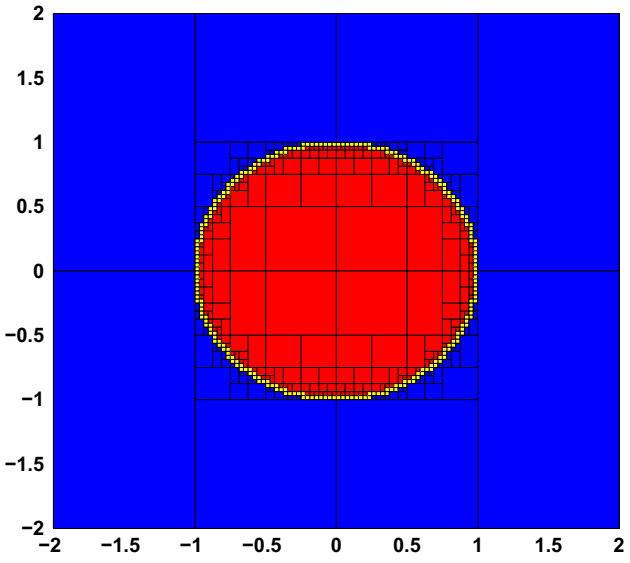


Fig. 2 The interior of a circle approximated by SIVIA

that this approximation was computed with $\varepsilon = 0.05$ for the approximation threshold used by SIVIA which corresponds to the yellow area. What is important to note in this Figure is that application of SIVIA *forms boxes that become smaller and smaller as they approach the circumference of the circle, while it forms larger boxes to cover areas that are away from the boundaries*.

Finally, let us remind, here, that to the extent the MLP model constitutes a successful implementation of the classification function, the different classes are successfully discriminated and so SIVIA successfully defines the corresponding domains of validity as disjoint areas in the input space. The domain of validity for an MLP network trained on a classification problem with M classes is defined as the union of M domains of validity one for each class [1].

4 Detailed description of the proposed approach

As noted in the previous sections, the aim of the proposed approach is to repudiate the uncertainty introduced by cross-validation, when evaluating generalization, by adopting a deterministic computing scheme based on the exact level of classification decision, which is consistently projected on the input space in order to reliably define the area promoting the chosen level of classification decision. For a detailed description of the proposed approach, first we need to introduce and analyze the key concepts on which this deterministic method is built. There are three interrelated concepts which constitute the foundation of the proposed method. These concepts are:

- the level of the classification decision of the neural network,
- the classification error taken into account by the previous level, and,
- the domain of validity corresponding to the level of classification decision.

These concepts and their relation are analyzed in the following subsection.

4.1 Key concepts of the proposed approach

The level of the classification decision The maximum output rule [9], typically, used for classification decisions as well as for computing confusion matrices introduces uncertainty in the classification decision which may vary from one training-testing experiment to the other, especially, when patterns are infected with noise. This is clearly true when classes have largely overlapping distributions. To avoid this burden, we introduce a clear cut to the classification decision using what we call a β-cut

classification decision. So, we define the interval $[1-\beta, 1]$ which delimits the output activity of the network to a range of values for which we are confident that it contains valid classification decisions. This affects both the classification error but, also, the area of the input space activating this classification decision. In "Appendix A", we provide a number of experiments in order to illustrate this argument.

The value of β should be carefully selected in order to reflect the right level of classification decision. The right choice of β makes it possible for the interval $[1-\beta, 1]$ to enclose the majority of the output values. A realistic way to select the value of β relies on the histogram of the MLP output values. The value of β, so considered, is the one which delimits the interval with the HPD for the MLP output. This choice reveals the relation of the β-cut classification decision with the credible region of the corresponding class variable ω. We consider that this point merits to be carefully studied and so it is left outside the work of this paper, where the value of β is manually specified, by simple observation of the output values histogram.

The classification error Defining a β-cut classification decision affects the classification error considered for the neural classifier. Depending on the quality of the training process, small values of β reduce the uncertainty of the classification decision but increase the classification error, while on the contrary, large values of β tend to diminish the classification error but hide the uncertainty induced by the classification decision due to incomplete training or other reasons.

Assumptions of the previous paragraph on the choice of β with regard to the HPD of the network seems to be a realistic solution. However, such a choice of β is not suitable when one compares two trained networks as it promotes the idiosyncrasies of each network. We believe that a more realistic and objective solution is to choose the smallest value among all the β's exhibited by different outputs of the networks.

The domain of validity corresponding to the level of classification decision Defining the domain of validity of a network classifier corresponding to a level for the classification decision, results in determining the area of the input space producing output values of the network classifier above the level of classification decision. For some specific class, this area includes all the parts of the input space attributed by the network to this class. This area comprises, also, the parts corresponding to both interpolation and extrapolation as performed by the network classification function.

4.2 Generalization evaluation based on the domain of validity

In the context of the proposed approach, an MLP classifies correctly a pattern into class, say C_j, if the value of the jth output is in the interval $[1-\beta, 1]$. In other words, the pattern at hand is correctly classified if it lies within the area of the domain of validity corresponding to the respective class. Therefore, the classification rule applied is: *the network classifies those patterns that fall into the area of the input space learned by the network during training*. This area is, precisely, the domain of validity of the network as defined by the IA-based inversion of the network. The above statement covers, also, classification of previously unseen patterns.

So, the extent to which the domain of validity covers the input space defines the area effectively classified by some neural classifier no matter if this contains known or as yet unknown patterns. In conclusion, we may state that *generalization can be estimated by a quantity which is proportionally related to the size, i.e., the volume, of the domain of validity of a trained network*.

With the proposed approach, the domain of validity is defined as a set of axis-aligned boxes of varying size and density. These characteristics reflect the quality of the input data (noise and class distribution), and the outcome of the training process affected by under-training, over-training and the ability to separate the classes. In "Appendix B", we discuss these aspects providing a number of experiments together with Figures illustrating these considerations.

A detailed qualification of each part of the domain of validity is not within the scope of the proposed method and to the best of our knowledge, in the literature no method is reported for performing such a qualification. However, when computing the volume of the domain of validity, for each class, we take into account for each hyper-box the number of patterns correctly classified or misclassified in the hyper-box. Such a consideration gives an idea of whether a hyper-box is part of a class area properly speaking or it is the result of interpolation or extrapolation of the network. These aspects are discussed in more details in the following subsection.

4.3 The new measures for evaluating generalization

Based on the above, we put forward the conclusion that the larger the domain of validity, the bigger its volume and so the higher the probability for some unknown pattern to be in this area and be classified. Hence, the necessary condition for some unknown pattern to be classified by the

network is to lie within the domain of validity of its respective class.

For any output node, given the interval $[1-\beta, 1]$, the inversion using the SIVIA method produces an inner approximation of the domain of validity as a union of axis-aligned boxes. The sum of the volumes of these boxes gives the volume of the domain of validity of each class C_j which is denoted here by V_j. Hence, if $V_1, V_2, \ldots, V_M$ are the volumes computed for the M classes, the total volume of the domain of validity of the network is given by $V_{\text{net}} = \sum_{i=1}^{M} V_i$. If V_{input} denotes the finite volume of the input space, then the ratio $\frac{V_{\text{net}}}{V_{\text{input}}}$, can be considered as the probability for some arbitrary pattern to be classified by the network.

Note that this probability can be maximized by the trivial solution where the output is valid (i.e., in the interval $[1-\beta, 1]$) for the whole input domain. In this case, one should take into account that a large percentage of the training patterns would be misclassified. Despite the fact that this is the result of poor training and, normally, should not happen, in the proposed method we address this issue even in more usual cases of inexact training. Actually, what we expect is that the network has both a large domain of validity and minimum percentage of misclassified or unclassified patterns. If l denotes the number of misclassified or unclassified training patterns and P is the total number of training patterns, we define the following measure of network performance that quantifies these two requirements:

$$G_{\text{net}} = \frac{V_{\text{net}}}{V_{\text{input}}} - \frac{l}{P}. \tag{2}$$

Hence, G_{net} is maximized when the domain of validity V_{net} is large and the network correctly classifies most of the training patterns. On the contrary, it attains low values when the validity domain V_{net} is small and/or the network either classifies incorrectly or does not classify several patterns.

Based on the discussion on under-training and over-training, in "Appendix A", we claim that low values of G_{net} are indications of network under-training, i.e., either the network is not trained sufficiently or its architecture is smaller than required. Moreover, low values for G_{net} may indicate over-fitting. This happens when then network architecture is oversized for some problem and many of the hidden nodes are used to form small regions around isolated training patterns or to shape details of the classes. These cases cannot be effectively addressed by G_{net} which is considered as a macroscopic measure in the sense that it is based on the whole volume of the validity domain and the total number misclassified/unclassified patterns, without focusing on the local details in various regions of the validity domain.

While under-training can be easily identified during training, over-training needs to be more carefully considered especially in the case of large networks. For this issue, we need to characterize the ability of the network to cope with the classes and patterns at a local level which means that each box should be taken into account to the extent it contains correctly classified patterns. So, we propose a second measure, called E_{net}, which takes into account local information of the domain of validity, as it considers for each valid box both its volume and its performance (i.e., whether the training patterns inside a box are correctly classified or not). More specifically, E_{net} is computed as follows:

- for each pattern $\mathbf{x}_n$ in the training set $\mathbf{X}$ let $C(\mathbf{x}_n)$ denote the class of this pattern,
- if C_k, $1 \leqslant k \leqslant M$, is the kth class then the following set of boxes is defined by IA-based inversion of the kth output node of the MLP: $\mathbb{B}^k = \left\{\boldsymbol{B}_1^k, \boldsymbol{B}_2^k, \ldots, \boldsymbol{B}_{N_k}^k\right\}$,
- let V_i^k denote the volume of $\boldsymbol{B}_i^k$,
- let X_i^k be the set of patterns $\mathbf{x}_n$ found to be inside the box $\boldsymbol{B}_i^k$.

For each box $\boldsymbol{B}_i^k$ we compute the local measure E_i^k:

$$E_i^k = \left[\sum_{\mathbf{x}_n \in X_i^k} \left(\mathbb{1}_{\mathbb{C}}(C(\mathbf{x}_n) = C_k) - \mathbb{1}_{\mathbb{C}}(C(\mathbf{x}_n) \neq C_k)\right)\right] V_i^k, \tag{3}$$

where $\mathbb{1}_{\mathbb{C}}$ is a suitable indicator function. Then the measure E_{net} is obtained by summing all E_i^k:

$$E_{\text{net}} = \sum_{k=1}^{M} \sum_{i=1}^{N_k} E_i^k. \tag{4}$$

This measure tends to consider hyper-boxes based on the number (density) of correctly classified patterns. This has the following effects:

- It deals with overlapping between classes. Actually, hyper-boxes covering areas where classes happen to overlap are included in some class to the extent they contain patterns of this class. Otherwise, they are not counted.
- It rejects regions of the domain of validity that do not contain any pattern at all. These areas of interpolation–extrapolation are taken into account by G_{net} but they do not contribute to the local behavior of the decision surface between classes.

– It favors the volume of hyper-boxes with high classification performance, i.e., containing many correctly classified training patterns.

A side effect of the last property is that the E_{net} measure provides lower values in the case of over-training. Actually, in the case of over-training, due to the existence of several small regions, the validity domain contains a higher number of low volume hyper-boxes. Therefore, even if the number of wrongly classified patterns is small, usually these patterns are included in low volume hyper-boxes. In such a case, each hyper-box contains a small number of patterns. Consequently, for a network that has been over-trained, the value of E_{net} is expected to be lower compared with the case a well-trained network. In general, it can be considered that E_{net} acts as a complementary metric to the first measure G_{net}.

In order to make the above arguments clear, we propose to observe the values of G_{net} and E_{net} related to Figs. 9 and 10 in "Appendix B". Figure 9 illustrates the domain of validity for the networks 2–4–2, 2–2–2 and 2–25–2. By simple visual inspection, one is able to verify that the network 2–2–2 cannot tackle this problem. Visual comparison of the networks 2–4–2 and 2–25–2 shows that the first one exhibits higher generalization as it is able to classify a larger area of the input space. Respective values for these figures are those found in Table 1. Moreover, one is able to compare the values of the measures G_{net} and E_{net} found in Table 1 and the values of the same network architectures trained on the same dataset, affected with noise, as they are displayed in Table 3.

In order to provide a unique metric of the generalization ability of a trained MLP, we propose to combine the two metrics G_{net} and E_{net} into a single one. The formula for computing this new metric needs to comply with the following arguments:

Table 1 Results for the two-dimensional artificial dataset (Higher values indicate better generalization)

Network model	Training error	G_{net}	E_{net}	M_{net}
2–2–2	0.0892	0.1904	6.4110	0.1904
2–3–2	0.0294	0.7574	13.8841	0.7574
2–4–2	**0.0100**	**0.9700**	**11.4154**	**0.9700**
2–5–2	9.9767e−05	0.9284	7.6616	0.9284
2–6–2	6.8297e−05	0.9226	10.4967	0.9226
2–8–2	9.9382e−05	0.8661	7.6610	0.8661
2–10–2	4.6347e−05	0.8960	11.6237	0.8960
2–15–2	5.3992e−05	0.8611	10.1423	0.8611
2–20–2	7.9098e−05	0.8823	7.2959	0.8823
2–25–2	4.9104e−05	0.8693	6.8902	0.8693

"According to its derivation rule, E_{net} is a quantity which takes values in the interval $[-V_{net}, V_{net}]$, where V_{net} denotes the total volume of the domain of validity. It is obvious that a zero or negative values of E_{net} denote that the network fails to correctly classify the input patterns, while positive values indicate that the majority of volumes of the hyper-boxes considered by E_{net} contain correctly classified patterns. While the role of E_{net} is to quantify these qualitative characteristics of the network classification function G_{net} is totally unaware of these subtleties and so it needs to be modified to reflect this situation as it is depicted by E_{net}."

In Eq. 5, we propose a formula for effectively combining G_{net} and E_{net} defining a new measure, called M_{net} as the product of G_{net} with the hyperbolic tangent (tanh) of E_{net}. The function 'tanh' is used to transform the E_{net} values (which could be either positive or negative) in the interval $[-1, 1]$, so that they are of the same scale as the G_{net} values. Note that M_{net} can take values between -1 and 1. In the case where E_{net} is negative, wrong classifications are expected to be higher than correct ones, thus the network is poorly trained. In this case the value of M_{net} is also negative indicating an unacceptable network. In the case where E_{net} is positive, $\tanh(E_{net})$ is also positive, thus M_{net} value is greater than zero and its value is high when both G_{net} and E_{net} measures are high. The results obtained in the experiments indicate that M_{net} constitutes an effective single measure for assessing generalization.

$$M_{net} = G_{net} \tanh(E_{net}). \quad (5)$$

5 Experimental evaluation and discussion

In this section, we describe a number of experiments carried out on artificial and real-world datasets in order to support the statements elaborated in the previous section. Then, we discuss the results obtained pointing out the characteristics supporting our hypotheses as well as some issues needed to be further considered in future work.

5.1 Experimental setup and evaluation

For the experimental evaluation of the proposed approach, six different datasets were used; two of them are artificial and the rest are real-world datasets found in the Machine Learning Repository of the University of California Irvine. The experiments with the two artificial and the Fisher-Iris datasets were executed using the SCS Toolbox [30] a package developed by Tornil-Sin and Puig at the Technical University of Catalonia in order to implement in Matlab the functionality related to SIVIA. For the interval

computations, the SCS Toolbox relies on INTLAB, the Matlab library of S. Rump [26] for interval computations.

The experiments with the other three datasets, namely, Bank Note Authentication, Vertebral Column (2 classes dataset) and Seeds, which are real-world problem, were carried out using IBEX, the C++ Library for interval computations, developed by Gilles Chabert et al. and released under the GNU Lesser General Public Licence (http://www.ibex-lib.org/). For all experiment results are shown in respective Tables 1, 2, 3, 4, 5, 6, 7, 8, where the best network architecture and related results are noted with boldface.

For all the experiments, the following disposition was retained: for each MLP used in the experiments, after training, a suitable natural inclusion function was written as a Matlab script function, or a C++ one. These functions were used by the specific SCS and the IBEX implementations of SIVIA, respectively. Moreover, for all experiments, the volume of the input space is the volume of the axis-aligned hyper-box defined by the size of each feature in the training data i.e.,

$$V_{\text{input}} = \prod_{i}^{N} |x_i^{\max} - x_i^{\min}|. \tag{6}$$

The first dataset is an artificial dataset for the classification of 200 patterns belonging to one of two classes, originally defined in [28]. The distribution of each class is a mixture of Gaussian designed in order to increase the difficulty in separability between the clusters formed by the patterns. The obtained dataset, presented in Fig. 3, is perfectly balanced as it accounts 100 patterns per class.

5.1.1 First experiment

For this experiment, we used the above dataset together with 10 MLPs having an architecture 2–H–2 with H taking on the values 2, 3, 4, 5, 6, 8, 10, 15, 20, 25, and using the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. The MLP networks were trained on the whole dataset using the Levenberg–Marquardt algorithm, and the proposed metrics were applied on the trained MLPs. The values of the metrics computed are shown in Table 1. In this experiment, the domain of validity is defined for each network for the interval [0.9, 1] meaning that the more appropriate value for having valid output values from the network was estimated to be $\beta = 0.1$.

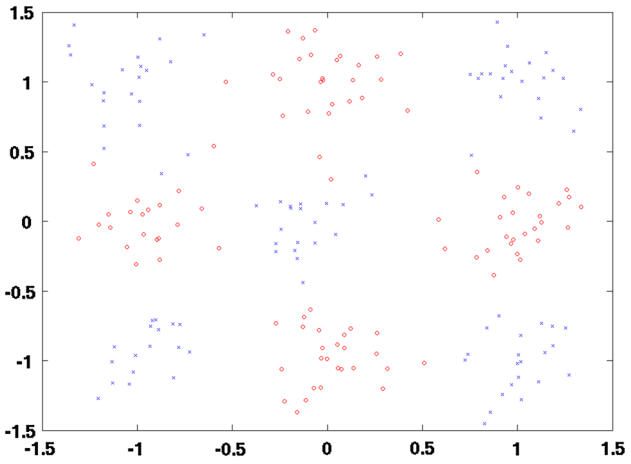


Fig. 3 The 2 dimensional artificial dataset

According to the results presented in Table 1, it seems that the MLP having the best generalization for the two-dimensional Artificial Dataset is the one with architecture 2–4–2. However, by performing a second round of training of the same MLP networks on the same problem using a different training algorithm, that is the scaled conjugate gradient (SCG), we see that it is not the same network having the best generalization performance.

More precisely, in Table 2, we present the training error obtained for this second training round and the values for the proposed metrics. The valid output interval for computing the validity domains was, here, set to [0.9, 1] and the bisection threshold parameter of the SIVIA algorithm used is the same as in the previous trial. Notice that in this case the MLP with the best generalization performance is the network 2–5–2 which proves to be slightly better than the network 2–10–2. Another point that needs to be commented is that the comparison of respective values in Tables 1 and 2 shows that even for the same networks, depending on the training algorithm the domains of validity can have different size and, so, the networks have different generalization ability.

5.1.2 Second experiment

This experiment was executed with the same dataset, as previously, which was tampered with in order to introduce noise in the form of badly placed patterns. This was done by changing the class label of 18 randomly selected

Table 2 More results for the two-dimensional Artificial Dataset (Higher values indicate better generalization)

Network model	Training error	G_{net}	E_{net}	M_{net}
2–2–2	0.1136	0.0587	6.3334	0.0587
2–3–2	0.0345	0.7477	12.1897	0.7477
2–4–2	0.0227	0.7923	10.7593	0.7923
2–5–2	**0.0055**	**0.9066**	**15.0557**	**0.9066**
2–6–2	9.9001e−05	0.8984	11.5352	0.8984
2–8–2	9.9409e−05	0.8660	12.6804	0.8660
2–10–2	9.9635e−05	0.9257	11.6573	0.9257
2–15–2	8.8107e−05	0.8846	8.9206	0.8846
2–20–2	9.1273s−05	0.8547	7.9647	0.8547
2–25–2	9.0279e−05	0.8636	7.7780	0.8636

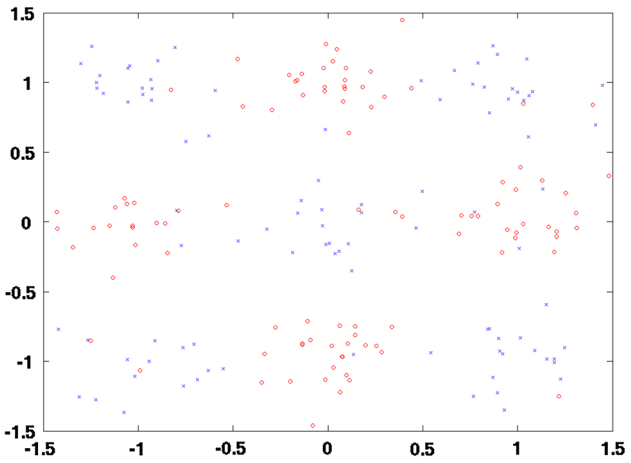


Fig. 4 The two-dimensional dataset tampered with by noise on the class labels

patterns over the 200 of both classes and resulted in the dataset shown in Fig. 4. Again, a set of MLPs of the same architecture 2–H–2, as previously, with H taking on the values 2, 3, 4, 5, 6, 8, 10, 15, 20, 25, 30, 35, 40 were trained on the whole dataset and the metrics G_{net}, E_{net} as well as the compound one M_{net} were computed. The results obtained are presented in Table 3. The value of β for this experiment was set to 0.1 and so the domain of validity for each network was computed for the interval [0.9, 1].

One may notice, here, that the network 2–20–2 seems to dispose the largest validity domain and, hence, it seems to give the better generalization performance. This conclusion is further supported by the value of E_{net} giving the best value for M_{net}. It is important to note, here, that given this "weird" dataset, in order to study over-training all networks were trained without precaution of early stopping. This explains, somehow, why the network with the best performance is the one with 20 nodes and not the network with 8 nodes which demonstrates better behavior concerning over-training but covers significantly less part of the input space.

Table 3 Results for the Noisy two-dimensional Artificial Dataset (Higher values indicate better generalization)

Network model	Training error	G_{net}	E_{net}	M_{net}
2–2–2	0.1502	0.0172	2.3285	0.0169
2–3–2	0.0972	0.3285	4.4934	0.3284
2–4–2	0.0744	0.5732	9.7774	0.5732
2–5–2	0.0601	0.7986	8.5169	0.7986
2–6–2	0.0550	0.8187	3.4648	0.8171
2–8–2	0.0400	0.9031	14.5668	0.9031
2–10–2	0.0273	0.7644	4.0001	0.7639
2–15–2	0.0130	0.8724	11.1084	0.8724
2–20–2	**0.0100**	**0.9628**	**6.5710**	**0.9628**
2–25–2	8.2230e−05	0.9227	5.1409	0.9226
2–30–2	6.8882e−05	0.9326	3.9244	0.9319
2–35–2	4.2505e−05	0.9473	4.4260	0.9470
2–40–2	9.2041e−05	0.9283	3.2069	0.9253

5.1.3 Third experiment

This experiment was carried out using a three-dimensional artificial dataset consisting of 3 classes. The distribution of each class is a Gaussian and the dataset consists of 500 patterns which is unbalanced as 200 belong to class 1, 100 belong to class 2 and the last 200 are taken from class 3. Figure 5 displays the dataset in a three-dimensional plot. The networks used in this experiment are also MLPs having an architecture 3–H–3 with H taking on the values 3, 4, 5, 6, 8, 10, 15, 20, 25, and using the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. The networks were trained with the Levenberg–Marquardt algorithm until reaching a training error set to 0.001 or for a maximum number of 2000 epochs. By simple inspection of the histogram of the output values for these networks, we concluded that the interval [0.9, 1] should be used by SIVIA in order to define the validity domains. The values of the metrics G_{net}, E_{net} and M_{net} computed for this experiment are shown in Table 4 below. In this experiment, we notice that the network 3–5–3 gives the best score for both measures G_{net} and E_{net} and therefore its value for M_{net} is the best among the networks in this experiment. So, we may conclude that this network gives the best generalization for this training dataset.

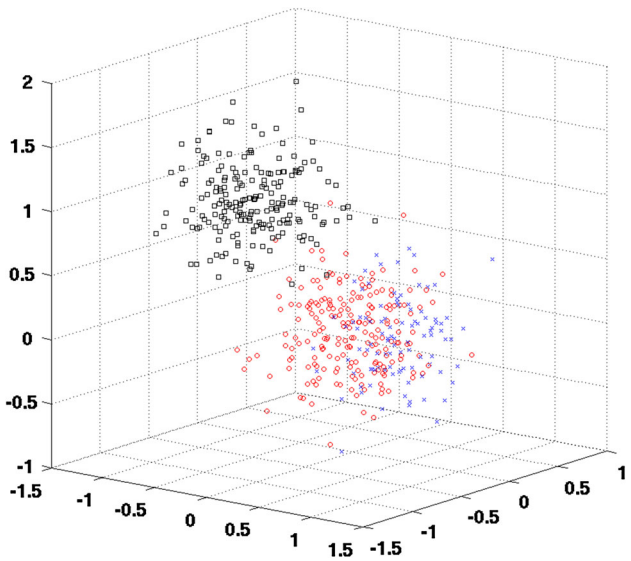


Fig. 5 The three-dimensional dataset with three classes

Table 4 Results for the three-dimensional Artificial Dataset (higher values indicate better generalization)

Network model	Training error	G_{net}	E_{net}	M_{net}
3–3–3	0.0093	0.8646	17.8032	0.8646
3–4–3	0.0080	0.9000	29.1629	0.9000
3–5–3	**0.0067**	**0.9248**	**56.1068**	**0.9248**
3–6–3	0.0020	0.8561	33.1453	0.8561
3–8–3	0.0027	0.8444	9.9164	0.8444
3–10–3	0.0040	0.8864	40.0438	0.8864
3–15–3	0.0027	0.8925	6.5828	0.8925
3–20–3	0.0013	0.8900	6.1050	0.8900
3–25–3	0.0020	0.8863	4.7675	0.8862

5.1.4 Fourth experiment

The dataset used in this experiment is the well-known Fisher-Iris problem. A number of 6 MLPs were trained on this dataset. These networks all have an architecture 4–H–3 with H taking on the values 2, 3, 5, 8, 10, 15, and they use the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. Each MLP was trained with the Levenberg–Marquardt algorithm until reaching an MSE of 0.001 or for a maximum number of 5000 epochs. The value retained for β is 0.2 and, so, the interval [0.8, 1] was used by SIVIA for the definition of the validity domains of the trained MLP networks. The results obtained for the proposed measures and the different MLP networks are summarized in Table 5.

5.1.5 Fifth experiment

For this experiment, we used the dataset of the Vertebral Column problem. In this problem, the biomedical data of 310 patients are used for two possible classification tasks. We retained the second task, where the categories Disk Hernia and Spondylolisthesis of the first task are merged into a single category labeled as 'abnormal'. So, this task consists in classifying patients as belonging to class Normal (100 patients) or Abnormal (210 patients). We used 10 different MLPs with one hidden layer, that is the architecture 6–H–2 with H taking on the values 3, 4, 5, 6, 8, 10, 15, 20, 25, 30 and using the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. Each MLP was trained with the Levenberg–Marquardt algorithm in two different ways one using a validation set with early stopping to avoid over-training and a second one without any precaution for over-training that is until the output error reaches an MSE of 0.001 or for a maximum number of 5000 epochs. The value retained for β is 0.2 and, so, the interval [0.8, 1] was used by SIVIA for the definition of the validity domains of the trained MLP networks.

Table 5 Results for the Fisher-Iris Dataset (Higher values indicate better generalization)

Network model	Training error	G_{net}	E_{net}	M_{net}
4–2–3	0.0040	0.4492	0.3303	0.1432
4–3–3	**2.2820e−04**	**0.7216**	**3.5389**	**0.7204**
4–5–3	5.0900e−04	0.5483	1.2730	0.4686
4–8–3	3.1881e−04	0.6072	0.8468	0.4186
4–10–3	1.1542e−04	0.3781	0.5105	0.1778
4–15–3	2.7132e−04	0.4407	0.4828	0.1976

The results obtained for the proposed measures and the different MLP networks are summarized in Table 6. Based on these results, the following remarks can be done regarding the measures of the proposed approach:

- The measures clearly derive the best architecture for this classification problem. The difference in the choice of the network between the two training methods i.e., 3 or 4 nodes in the hidden layer is negligible.
- The values of E_{net} indicate that this measure takes into account the over-training effect for both training methods. This is especially true for the training without early stopping where over-training almost surely happens.
- The evolution of the values of M_{net} for the case of training with validation set complies with the well-known rule (Occam's razor), for sufficiently trained network, the most suitable architecture in terms of generalization is that one with less nodes in the hidden layer.
- It is obvious that he previous remark is not valid when training of a network is done without over-training precautions.

5.1.6 Sixth experiment

For this experiment, we used the dataset of the Bank Note Authentication problem. This dataset contains the features of image data taken from "genuine and forged banknote-like specimens". The digitally processed images have 400×400 pixels, and the features are extracted using wavelet transform. The features are: variance, skewness, kurtosis and entropy of the wavelet transformed image. Based on the values of these feature, each image is classified as belonging to a genuine or a forged banknote-like specimen. The classes have 762 and 610 patterns. We trained 10 different MLPs on this dataset. These networks

Table 6 Results for the Vertebral Column 2-classes Dataset

Network model	Results of training with validation set (early stopping)			Results of training without validation set (no early stopping)		
	G_{net}	E_{net}	M_{net}	G_{net}	E_{net}	M_{net}
6–3–2	0.7346	0.1676	0.1220	**0.7049**	**1.1717**	**0.5814**
6–4–2	**0.7450**	**1.1260**	**0.6032**	0.6068	0.0000	0.0000
6–5–2	0.7608	0.5441	0.3774	0.6203	0.0007	0.0004
6–6–2	0.6724	0.5106	0.3163	0.6191	0.0157	0.0097
6–8–2	0.7345	0.1760	0.1279	0.6159	1.1370	0.5010
6–10–2	0.6257	0.0639	0.0399	0.6983	0.2054	0.1415
6–15–2	0.6659	0.0089	0.0059	0.6746	0.0391	0.0264
6–20–2	0.6686	0.0200	0.0134	0.6400	0.0279	0.0179
6–25–2	0.7094	0.0222	0.0157	0.6012	0.0298	0.0179
6–30–2	0.6467	0.0289	0.0187	0.5538	0.0039	0.0022

all have an architecture 4–H–2 with H taking on the values 2, 3, 4, 5, 6, 7, 8, 9, 10, 15 and they use the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. Each MLP was trained with the Levenberg–Marquardt algorithm with early stopping to avoid over-training setting for the output error an MSE of 0.001 and a maximum number of 5000 epochs. The value retained for β is 0.2 and, so, the interval [0.8, 1] was used by SIVIA for the definition of the validity domains of the trained MLP networks.

The results obtained for the proposed measures and the different MLP networks are summarized in Table 7. Based on these results, we deduce that the network 4–4–2 is the most efficient in terms of generalization for this specific problem with the classification decision level set to $\beta = 0.2$. Both inefficient and overqualified network architectures are revealed by the values of the proposed measures. As in the previous experiment, the evolution of the values of M_{net} complies with the well-known rule (Occam's razor), for sufficiently trained network the most suitable architecture in terms of generalization is that one with less nodes in the hidden layer.

Table 7 Results for the Bank Note Authentication Dataset

Network model	G_{net}	E_{net}	M_{net}
4–2–2	0.5710	9.0786	0.5710
4–3–2	0.7106	2.4860	0.7008
4–4–2	**0.7943**	**34.2739**	**0.7943**
4–5–2	0.6470	13.2994	0.6470
4–6–2	0.7354	9.2867	0.7354
4–7–2	0.7225	11.4105	0.7225
4–8–2	0.6033	18.9183	0.6033
4–9–2	0.6095	2.2829	0.5970
4–10–2	0.6087	0.5761	0.3164
4–15–2	0.6070	0.6485	0.3464

5.1.7 Seventh experiment

For this experiment, we used the dataset of the Seeds problem. This dataset concerns the classification of kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian. The dataset consists of 70 elements of each class. Seven geometric parameters of wheat kernels were measured and are the features of each pattern. These geometric parameters are: area (A), perimeter (P), compactness ($C = 4 * pi * A/P^2$), length of kernel, width of kernel, asymmetry coefficient and length of kernel groove.

We trained 8 different MLPs on this dataset. These networks all have an architecture 7–H–3 with H taking on the values 2, 3, 4, 5, 6, 7, 8, 10 and they use the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. Each MLP was trained with the Levenberg–Marquardt algorithm with early stopping to avoid over-training setting for the output error an MSE of 0.001 and a maximum number of 5000 epochs. The value retained for β is 0.2 and, so, the interval [0.8, 1] was used by SIVIA for the definition of the validity domains of the trained MLP networks.

The results obtained for the proposed measures and the different MLP networks are summarized in Table 8. Based on these results, we deduce that the network 7–5–3 is the most efficient in terms of generalization for this specific problem with the classification decision level set to $\beta = 0.2$. The most impressive in these results concerns the very small values of E_{net} for all networks. The explanation we may give is that this is a sparse dataset with only 210 patterns dispersed in three classes. However, the values of M_{net} are representative of the behavior of the networks for this classification problem. Again the values of M_{net} seem to comply with the rule that for sufficiently trained network the most suitable architecture in terms of generalization is that one with less nodes in the hidden layer.

Table 8 Results for the Seeds Dataset

Network model	G_{net}	E_{net}	M_{net}
7–2–3	0.5465	0.0001	0.0001
7–3–3	0.4727	0.0538	0.0254
7–4–3	0.8795	0.0168	0.0148
7–5–3	**0.8954**	**0.0346**	**0.0310**
7–6–3	0.9102	0.0144	0.0131
7–7–3	0.8434	0.0101	0.0085
7–8–3	0.8648	0.0077	0.0066
7–10–3	0.8287	0.0072	0.0060

5.2 Comparison with *K*-fold cross-validation

Though a comparison of the proposed approach with cross-validation is not relevant in the sense that these two approaches aim to assess generalization of neural networks from completely different points of view and stages of development, here we present the results on two experiments we designed and carried for our previous work [2] in a specific context. In this context, the proposed approach is applied to every fold set for cross-validation and the final values of the measures are averaged over the values obtained on the different folds. These results were verified and minor corrections were made. Hereafter, we offer a more detailed analysis of the results.

The first experiment concerning the comparison of the proposed approach against *K*-fold cross-validation uses the two-dimensional artificial dataset with 200 patterns as in Sect. 5.1, above. *K*-fold cross-validation was applied to this dataset with $K = 20$, thus, giving a number of 20 training trials with 19 subsets while holding out one and using it as test subset. A set of 8 MLPs was used having an architecture 2–H–2 with H being successively 4, 5, 6, 8, 10, 15, 20, 25, and having the hyperbolic tangent for all nodes in the hidden layer and the logistic sigmoid for the output nodes. Each MLP was trained with the Levenberg–Marquardt algorithm until reaching a MSE of 0.001 or for a maximum number of 5000 epochs thus giving a total number of $(20\,\text{trials} \times 8\,\text{models}) = 160$ training trials.

It is worth noting here that for each trial (a network architecture trained on 20 folds) the generalization was computed as the average of the correctly classified test patterns for the cross-validation method. Moreover, the values of the proposed measures, G_{net} and E_{net}, were computed with respect to the *K*-fold splitting. This means that for each network after training it with $K - 1$ folds the trained network was inverted in order to determine its validity domain. As a result, we obtained 20 different domain of validity and we computed the corresponding values for the measures G_{net} and E_{net}, while the final values for these measures are computed by averaging the 20 values of the folds. The interval of the network output values inverted using SIVIA was [0.8, 1]. Note, that in this experiment, only the test patterns were used for determining whether a hyper-box should take part in the E_{net} sum. This was done in order to be inline with cross-validation and the *K*-fold splitting of the dataset.

Table 9 outlines the results of this experiment. In order to facilitate comparison with the values computed by *K*-fold cross-validation, the values of G_{net} are multiplied by a factor of 100 and they are reported in the form of percentage. One may notice that the generalization measures provided by the proposed approach are consistent with theory i.e that generalization degrades as the number of hidden units, i.e., the complexity of the MLP increases. Obviously, this is not the case with the values computed by *K*-fold cross-validation.

The second experiment presented, here, for the comparison of the proposed approach with *K*-fold cross-validation, using the Fisher-Iris dataset. This experiment, also, was designed and presented in [2] and the verification of the result did not reveal any difference with initial ones. The dataset was divided in ten subsets in order to compare the results obtained by the proposed approach with the multifold (10-fold) cross-validation. A set of 6 MLPs were used and they all have the same architecture 4–H–3 with H being successively 2, 3, 5, 8, 10, 15. All MLPs use the hyperbolic tangent for the nodes in their hidden layers and the logistic sigmoid for their output nodes. They are trained with the Levenberg–Marquardt algorithm until reaching an MSE of 0.001 or for a maximum number of 5000 epochs.

Concerning cross-validation, for each fold the generalization performance is computed as the rate of the correctly classified test patterns and the total of 10 values are averaged to provide a mean value for the generalization of the corresponding network. Regarding the proposed approach, the measures G_{net} and E_{net} were computed following the same computing scheme as in the previous experiment. This turned out to give for each trained network 10 different domains of validity and respective values for the measures which are averaged to obtain the final ones shown in Table 10. The interval of the network output values inverted using SIVIA was [0.8, 1]. Again, in order to facilitate comparison with the values computed by *K*-fold cross-validation, G_{net} is here multiplied by a factor of 100 and its values are given in the form of percentage. Note, that as in the previous experiment only the test patterns were used for determining whether a hyper-box should take part in the E_{net} sum. This was done in order to be inline with cross-validation and the *K*-fold splitting of the dataset.

Table 9 Comparison of the proposed approach with K-fold cross-validation for the two-dimensional artificial dataset

Network model	Cross-validation mean (std)	G_{net} mean (std)	E_{net} mean (std)
2–4–2	90.50% (9.45%)	96.45% (2.81%)	15.80 (16.65)
2–5–2	91.50% (8.75%)	94.50% (2.76%)	10.44 (7.69)
2–6–2	91.00% (9.68%)	91.35% (1.72%)	9.98 (4.36)
2–8–2	93.00% (8.65%)	90.63% (2.64%)	11.28 (4.62)
2–10–2	93.50% (6.71%)	88.84% (1.54%)	9.51 (1.76)
2–15–2	92.00% (8.34%)	89.08% (2.10%)	9.87 (1.23)
2–20–2	93.00% (8.65%)	88.38% (1.35%)	8.45 (0.76)
2–25–2	93.00% (7.33%)	87.65% (1.07%)	7.49 (0.80)

Table 10 Comparison of the proposed approach and K-fold cross-validation for the Fisher-Iris Dataset

Network model	Cross-validation mean (std)	G_{net} mean (std)	E_{net} mean (std)
4–2–3	94.67% (6.13%)	80.42% (10.58%)	2.14 (4.09)
4–3–3	94.00% (6.63%)	79.66% (7.0%)	1.69 (1.28)
4–5–3	93.33% (7.70%)	73.59% (6.08%)	1.15 (1.05)
4–8–3	91.33% (8.92%)	67.67% (6.68%)	1.01 (0.74)
4–10–3	94.67% (10.33%)	66.98% (10.04%)	0.84 (0.75)
4–15–3	92.67% (7.98%)	63.91% (8.49%)	0.51 (0.51)

5.3 Discussion

Following the previous experiments, a number of issues can be discussed concerning the analysis and the assumptions in this paper. First, one should note that the values of G_{net}, E_{net} and M_{net} provide relevant indicators concerning the generalization ability of the networks used in the experiments as they reflect the quality of the trained networks in terms of generalization. This is true, especially, concerning the values of the measures in the cases of under-training and over-training as well as the compliance of the values observed with the rule relating the generalization ability of a network with its complexity.

The values obtained in all experiments clearly depend on the level of classification decision (β) set but also on the complexity of the network and the outcome of its training. As noted in Sect. 4.1, choosing the right value for β is important in defining the domain of validity corresponding to a classification decision with minimal uncertainty. A choice based on the HPD of the network output (any output) constitutes a reasonable choice which merits to be further investigated in future work under the assumption that the resulting domain of validity corresponds to a credible set defined for the value of β. However, for any choice of β its value extends or retracts the corresponding domain of validity affecting mainly G_{net} and to a much lesser extent E_{net} which depends on local information of the domain of validity. Finally, it goes without saying that if one needs to compare generalization of two network architectures on the same problem then one has to choose the same value of β for both networks.

When comparing the proposed approach with K-fold cross-validation, we need to note that, in contrast with this widespread technique, the proposed approach needs no test set for assessing generalization and takes into account the whole dataset. Even in the case of networks trained on parts of the dataset as in the case of the experiments comparing the proposed approach with K-fold cross-validation Tables 9 and 10 show that the values of the metrics G_{net}, E_{net} and M_{net} are more representative of the generalization ability of the networks. Actually, "small" MLPs give lower values for these measures, thus, indicating under-training, until some "critical" architecture for which the values of the metrics are higher and take peak values, meaning that this architecture best fits the problem at hand. Then, for networks with greater number of hidden units, the values of these metrics, despite local fluctuations, decrease, thus, permitting to conclude that they are consistent with the theoretic statement that "large" networks are prone to over-training.

A close look of the results presented in Tables 1 and 2 shows that the same network architecture may give different validity domains for different training algorithms. Hence, by simply calculating the classification error at the end of training, one is not able to draw consistent conclusions about the generalization ability of a network. The proposed measures can tackle this important issue since they permit to compare, in terms of generalization, the performance of networks having the same architecture, but trained with different algorithms for the same problem. Note that this kind of comparison is outside the scope of K-fold cross-validation.

Another issue to be discussed is related to the value of the parameter ε used by SIVIA during network inversion for controlling bisection of the hyper-boxes. In our experiments the typical value used for this parameter is $\varepsilon = 0.01$. This specific value seems to be the most suitable for the experiments carried out as it provides significant results with respect to the minimal distance between patterns in the input space. For some problems, if a smaller value seems to fit better, one should take into account the increase of the computation cost this implies given that the exact determination of the domain of validity requires an exhaustive analysis of the input space. The tradeoff between the value of ε and the computational cost of the approach is another issue requiring further investigation and certainly deals with the well-known class separability measures in Machine Learning [28]. Obviously, this issue constitutes an open question for future study.

Finally, a matter that needs to be pointed out here and certainly constitutes a task for future investigation is the definition of a concise mathematical model supporting the proposed approach and the results obtained. For the time being, we underline that the estimation of generalization, as proposed herein, is a reliable result of a deterministic approach.

6 Conclusion

In this paper, we proposed a new approach for estimating the ability of a neural network to generalize on a classification problem it was trained. Based on a concrete classification decision for the network outputs, the approach defines the domain of validity of the trained network. The domain of validity is defined to be the part of the input space providing confident network outputs, i.e., outputs in a interval whose lower bound is the level of the classification decision. The domain of validity is computed by inverting the network output applying an IA-based inversion method which guarantees the accuracy of the size and the position of this area in the input space. Given these consistent characteristics, one is able to derive empirical metrics of the network classification performance in terms of generalization which comply with theoretical considerations.

The proposed approach contributes to the estimation of neural network generalization by considering its domain of validity and without recourse to any kind of test set. Thus, the entire dataset can be used for training, there is no need for separate testing and so, two different network architectures can be compared in terms of generalization in a deterministic way by considering the area of the input space, effectively seen by each architecture. On the contrary, the widespread K-fold cross validation is not able to derive such an assessment of generalization and it is used to statistically estimate the generalization ability of some network architecture after having performed a significant number of training experiments. The validity of the assumptions, formulated in this paper, and the reliability of the techniques used support our confidence regarding the potential of the proposed approach. The measures derived are strongly supported by the results of the experiments we carried out on various artificial and real-world problems. When compared against cross-validation, the proposed approach provides results which are consistent with the theory.

We estimate that a side effect of the proposed approach is that it offers the possibility to study, via the network classification function, the distribution of the data in the input space, the vicinity of the different classes, their overlap and even the prospect to compute approximate values of the Hausdorff distance between the classes. Moreover, the domain of validity can be seen as a level set in the input space resulting from the inversion of an output interval. It is interesting to study, using IA-based inversion, the extent to which a trained network participates in forming such level sets and, hence, it performs a non-parametric estimation of the probability distribution function of the input data [19]. All these constitute open issues for future study.

The previous considerations deal, also, with the need to validate the proposed empirical metrics from a mathematical point of view which will take into account the Bayesian aspects of the neural network classification function. Finally, we are concerned by a suitable implementation of SIVIA for neural network inversion which should perform inversion incrementally, i.e., for each class exclude from the search space the area found to belong to previously examined classes. This will help to reduce the computational cost of applying a branch-and-bound technique to explore the entire input space.

Acknowledgements The authors would like to thank the anonymous reviewers for their valuable suggestions and comments on earlier version of the manuscript that helped to significantly improve the paper at hand.

Compliance with ethical standards

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

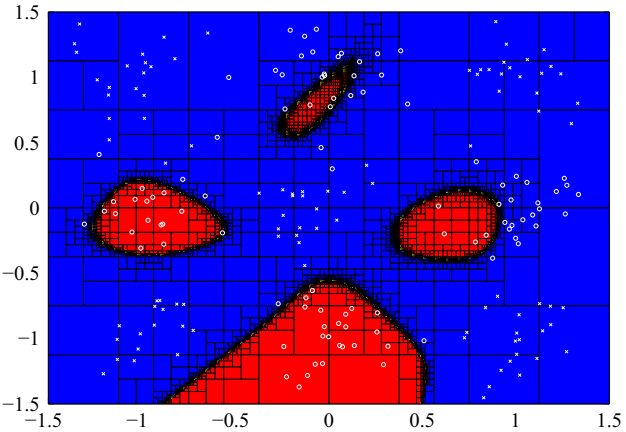
Appendix A

In order to illustrate the impact of the β-cut on the domain of validity first let us consider the two-dimensional classification dataset with two classes forming nine groups shown in Fig. 6a. A $2-10-1$ MLP, using logistic

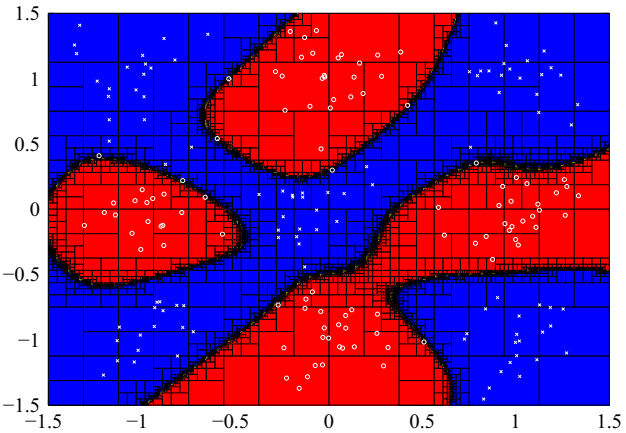
sigmoid activation functions, has been trained on this dataset, and the contour plot of its output is shown in Fig. 6b. In this Figure, the white regions (output greater than $1-\beta$) correspond to patterns classified by the MLP network into class 1 (red points), while the black regions (output lower than β) correspond to patterns classified into class 2 (blue points). Obviously, the gray level zone depicts the ambiguity of classification for patterns near the class boundaries and provides MLP output values in the interval $[\beta, 1-\beta]$.

The impact of this *β-cut classification decision* is better depicted in Figs. 7 and 8. For each one of these Figures, the red colored area corresponds to a specific domain of validity defined for some specific interval $[1-\beta, 1]$ of the network output, for the MLP trained on the above two-dimensional problem. Each area is determined using SIVIA to invert the MLP output interval $[1-\beta, 1]$ for class 1 in the input space.

It is obvious that the value of β clearly extends or restricts the input space area classified by the MLP into class 1. This argument can be easily verified by simple observation of Fig. 7a, b, while for problems with a higher dimension this can be confirmed with the comparison of the volumes of the respective domains of validity. This shows the importance of choosing the right value for β which, here, needs to be 0.1 if one wants to take the right classification decision for a significant part of the input space. As shown in Fig. 8a, b, the appropriate value of β also depends on the number of training epochs and the error threshold that were used chosen to train the MLP.



(a) When the output interval is [0.999,1]

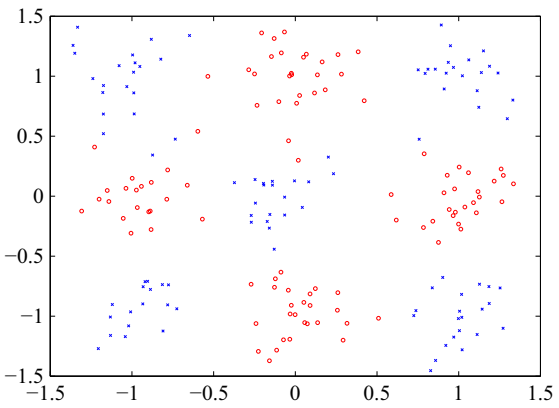


(b) When the output interval is [0.9,1]

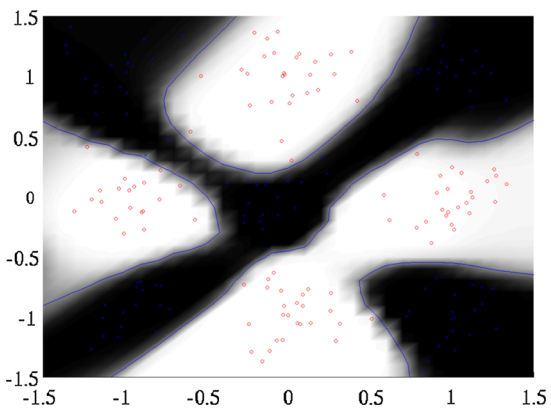
Fig. 7 The domain of validity for an MLP trained with 500 epochs and MSE $\leqslant 1e-03$

Appendix B

In the extreme case, of a pattern producing more than one valid outputs (i.e., it is assigned to more than one classes), the current implementation computing the domain of validity results in considering this pattern misclassified. Actually, for its proper class the pattern is correctly classified while for any other class for the other classes it is a misclassified pattern. A previous approach for determining

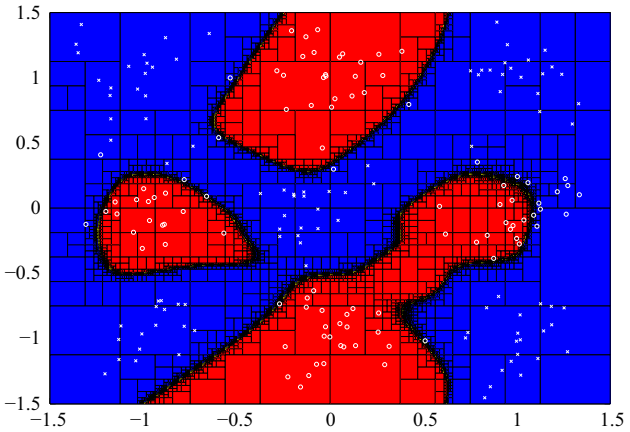


(a) The artificial dataset

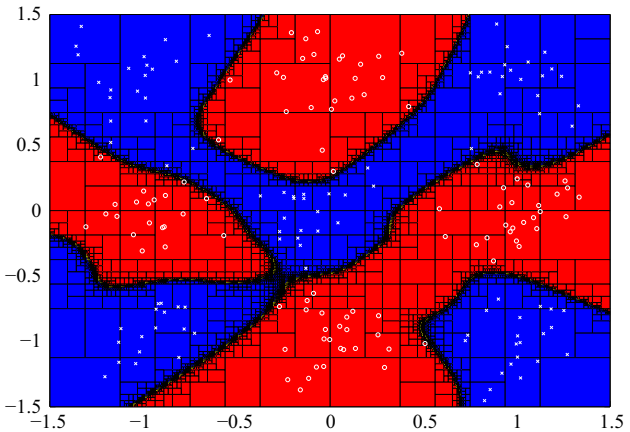


(b) Contour plot of the MLP output

Fig. 6 The artificial dataset and the contour plot of an MLP trained on this dataset



(a) When the output interval is [0.999,1]

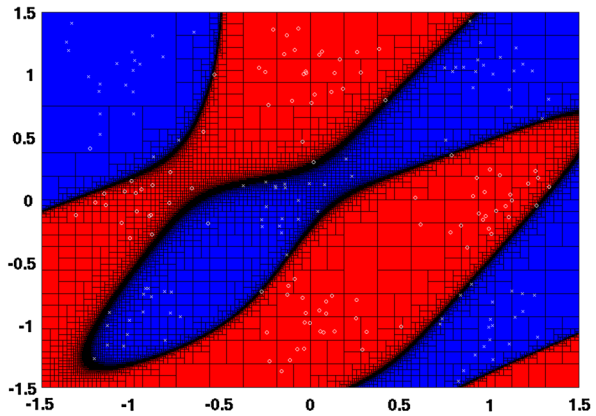


(b) When the output interval is [0.9,1]

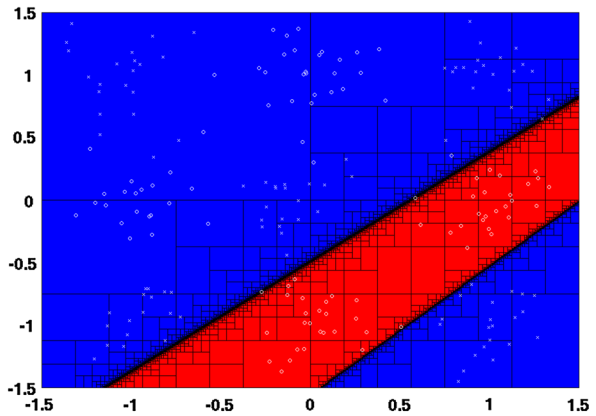
Fig. 8 The domain of validity for an MLP trained with 5000 epochs and MSE $\leqslant 1e-05$

the domain of validity considered such a pattern unclassified. However in terms of the proposed metrics both approaches compute the same result given that unclassified and misclassified patterns have the same status for the computed metrics.

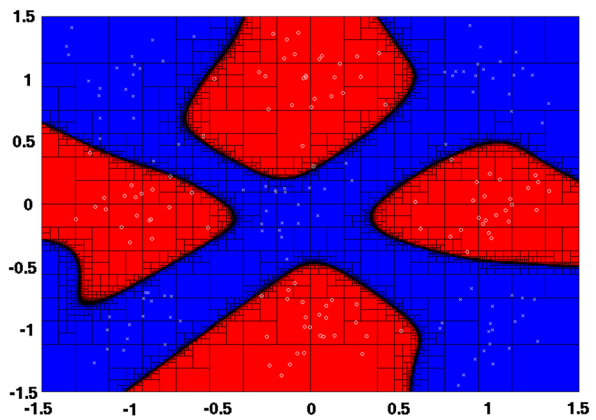
In many cases a training algorithm results in either under-trained or over-trained networks. Under-training arises for many reasons (insufficient training, small sized training data, inappropriate network architecture, etc.). In consequence, as shown in Fig. 9b, the domain of validity covers either small regions of the input space or a large region is incorrectly classified. For instance, the validity domain of a 2–4–2 MLP, shown in Fig. 9a, exhibits a more regular coverage of the input space, while the 2–2–2 MLP, as shown in Fig. 9b manages to cover a narrow strip in the input space. In general, it can be stated the validity domain of an under-trained network is composed of a small number of large regions with regularly shaped boundaries.



(a) Domain of validity upon successful training of a $2-4-2$ MLP



(b) Domain of validity following under-training of a $2-2-2$ MLP



(c) Domain of validity following over-training of a $2-25-2$ MLP

Fig. 9 Indicative examples of different domains of validity

Besides under-training, another issue affecting generalization is network over-training. Typically, an over-trained network fails to correctly classify unseen patterns as it has learned "exactly" the training data and hence it is not able to generalize well. In this case the decision boundaries computed by the network delimit as firmly as possible the regions in the input space and the network fails to

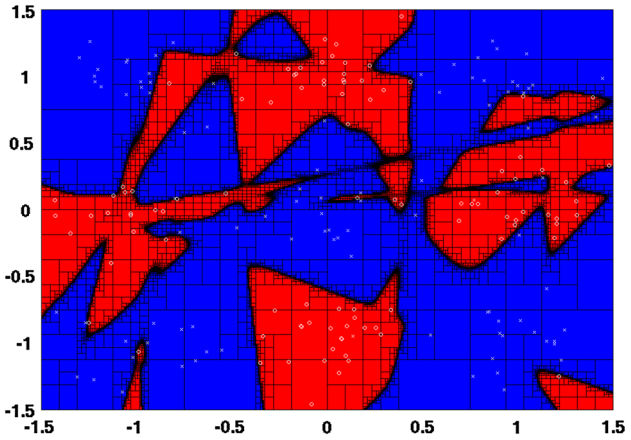


Fig. 10 Example of over-training for a 2–20–2 MLP trained with noisy data

interpolate even among close neighboring groups. In such a case the domain of validity consists of smaller regions and so its volume diminishes. An indicative example of such a validity domain is given in Fig. 9c. As a result we may state that for a well-trained network, the lower the volume of its domain of validity, the poorer the generalization achieved by the network due to over-training.

Another unfortunate result, when considering over-training is that MLPs, especially those with a high number of nodes in the hidden(s) layer(s), tend to fit outliers, noisy input patterns as well as patterns with noisy class labels, see Fig. 10. In these cases the network has the flexibility to form the decision boundaries that discriminate the outlying or misplaced patterns. Doing so, the network defines isolated regions, such as isles or lobes, in the input space which delimit not only these very patterns but also important parts of the input space for which there is no information about the class or the classes they belong to. In general, it can be stated that the validity domain of an over-trained network contains regions with small size and irregularly shaped boundaries.

Hence, the previous cases constitute different aspects of over-training that need to be taken into account when considering the volume of the domain of validity as a metric of the network's generalization performance.

References

1. Adam SP, Karras DA, Magoulas GD, Vrahatis MN (2015) Reliable estimation of a neural network's domain of validity through interval analysis based inversion. In: 2015 international joint conference on neural networks (IJCNN), pp 1–8. https://doi.org/10.1109/IJCNN.2015.7280794
2. Adam SP, Likas AC, Vrahatis MN (2017) Interval analysis based neural network inversion: a means for evaluating generalization. In: Boracchi G, Iliadis L, Jayne C, Likas A (eds) Engineering applications of neural networks. Springer International Publishing, Berlin, pp 314–326
3. Adam SP, Magoulas GD, Karras DA, Vrahatis MN (2016) Bounding the search space for global optimization of neural networks learning error: an interval analysis approach. J Mach Learn Res 17(169):1–40. http://jmlr.org/papers/v17/14-350.html
4. Bishop CM (1996) Neural networks for pattern recognition. Oxford University Press, Oxford
5. Courrieu P (1994) Three algorithms for estimating the domain of validity of feedforward neural networks. Neural Netw 7(1):169–174
6. Eberhart R, Dobbins R (1991) Designing neural network explanation facilities using genetic algorithms. In: 1991 IEEE international joint conference on neural networks, vol 2, pp 1758–1763
7. Hampshire II JB, Pearlmutter BA (1991) Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. In: Proceedings of the 1990 connectionist models summer school, vol 1, pp 159–172
8. Hassoun MH (1995) Fundamentals of artificial neural networks. MIT Press, Cambridge
9. Haykin S (1999) Neural networks a comprehensive foundation, 2nd edn. Prentice-Hall, Upper Saddle River, NJ
10. Hernández-Espinosa C, Fernández-Redondo M, Ortiz-Gómez M (2003) Inversion of a Neural Network via Interval Arithmetic for Rule Extraction. In: Kaynak O, Alpaydin E, Oja E, Xu L (eds) Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003, vol 2714. Springer, Berlin Heidelberg, pp 670–677 Lecture Notes in Computer Science
11. Jaulin L, Kieffer M, Didrit O, Walter E (2001) Applied interval analysis with examples in parameter and state estimation, robust control and robotics. Springer, London
12. Jaulin L, Walter E (1993) Set inversion via interval analysis for nonlinear bounded-error estimation. Automatica 29(4):1053–1064
13. Jensen C, Reed R, Marks R, El-Sharkawi M, Jung JB, Miyamoto R, Anderson G, Eggen C (1999) Inversion of feedforward neural networks: algorithms and applications. In: Proceedings of the IEEE 87(9):1536–1549
14. Kamimura R (2017) Mutual information maximization for improving and interpreting multi-layered neural networks. In: 2017 IEEE symposium series on computational intelligence (SSCI), pp 1–7
15. Karystinos GN, Pados DA (2000) On overfitting, generalization, and randomly expanded training sets. IEEE Trans Neural Netw 11(5):1050–1057
16. Kearfott RB (1996) Interval computations: introduction, uses, and resources. Euromath Bull 2(1):95–112
17. Kiefer J, Wolfowitz J (1952) Stochastic estimation of the maximum of a regression function. Ann Math Stat 23:462–466
18. Kindermann J, Linden A (1990) Inversion of neural networks by gradient descent. Parallel Comput 14(3):277–286
19. Likas A (2001) Probability density estimation using artificial neural networks. Comput Phys Commun 135(2):167–175
20. Liu Y (1995) Unbiased estimate of generalization error and model selection in neural network. Neural Netw 8(2):215–219
21. Lu BL, Kita H, Nishikawa Y (1999) Inverting feedforward neural networks using linear and nonlinear programming. IEEE Trans Neural Netw 10(6):1271–1290
22. Novak R, Bahri Y, Abolafia DA, Pennington J, Sohl-Dickstein J (2018) Sensitivity and generalization in neural networks: an empirical study. In: International conference on learning representations. https://openreview.net/forum?id=HJC2SzZCW
23. Reed R, Marks R (1995) An evolutionary algorithm for function inversion and boundary marking. In: IEEE international conference on evolutionary computation, 1995, vol 2, pp 794–797

24. Richard M, Lippmann R (1991) Neural network classifiers estimate Bayesian a posteriori probabilities. Neural Comput 3(4):461–483. https://doi.org/10.1162/neco.1991.3.4.461
25. Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22:400–407
26. Rump SM (1999) INTLAB - INTerval LABoratory. In: Csendes T (ed) Developments in reliable computing. Kluwer Academic, Dordrecht, Netherlands, pp 77–104
27. Saad EW, Wunsch DC II (2007) Neural network explanation using inversion. Neural Netw 20(1):78–93
28. Theodoridis S, Pikrakis A, Koutroumbas K, Kavouras D (2010) Introduction to pattern recognition: a MATLAB approach. Academic Press, Burlington, MA 01803, USA
29. Thrun SB (1993) Extracting provably correct rules from artificial neural networks. Technical Report IAI–TR–93–5, Institut fur Informatik III, Bonn, Germany
30. Tornil-Sin S, Puig V, Escobet T (2010) Set computations with subpavings in MATLAB: the SCS toolbox. In: 2010 IEEE international symposium on computer-aided control system design (CACSD), pp 1403–1408
31. Wolpert DH (1990) A mathematical theory of generalization: part I. Complex Syst 4(2):151–200
32. Wolpert DH (1990) A mathematical theory of generalization: part II. Complex Syst 4(2):201–249
33. Wolpert DH (1992) On the connection between in-sample testing and generalization error. Complex Syst 6(1):47–94
34. Wolpert DH (1996) The existence of a priori distinctions between learning algorithms. Neural Comput 8(7):1391–1420. https://doi.org/10.1162/neco.1996.8.7.1391
35. Wolpert DH (1996) The lack of a priori distinctions between learning algorithms. Neural Comput 8(7):1341–1390. https://doi.org/10.1162/neco.1996.8.7.1341