Three Algorithms for Estimating the Domain of Validity of Feedforward Neural Networks

PIERRE COURRIEU
Centre National de la Recherche Scientifique and Université de Provence

(Received 22 December 1992; accepted 23 July 1993)

Abstract—This article presents three simple algorithms for determining the distance between any point, and the domain of interpolation associated with a cluster of control points of a vectorial function. The first algorithm uses the convex hull polytope of the cluster in the support space to accurately estimate the domain. The second algorithm uses a neuron-like good approximation of the first. When the number of vertices of the polytope is large, a more economical approach is to approximate the domain by its circumscribed sphere, which is what the third algorithm does. It is also shown that there is a significant relation between these three measures of the distance between any test point and a set of learning points, and the generalization errors made by an artificial neural network.

Keywords—Interpolation domain, Neural networks, Convex polytope, Circumscribed sphere, Generalization reliability.

1. INTRODUCTION

Feedforward, continuous input/output neural networks can be viewed as a type of numerical smoother (i.e., filter-interpolator) of the set of learning points from which they were configured. It has been demonstrated that any nonlinear function can be approximated by neural networks with hidden units (Funahashi, 1989). One of the specific advantages of neural smoothers is that they place few restrictions on the structure of the set of control points (the learning points). Hence, it is not necessary to sample the interpolation support domain with a regular patch to structure the set of points, as is the case in other types of interpolators (for a review, see Böhm, Farin, & Kahmann, 1984). It is thus possible to use random learning sets or learning sets based on cognitive criteria that are independent of artificial calculation requirements. In addition, one avoids the exponential increase in the sample size when the dimension of the support space (i.e., the number of input variables) increases. The counterpart of this relative flexibility is that we generally do not know whether the values of the input variables sent to the neural network in a request for a generalization response will fall within the domain of interpolation associated with the learning set. Indeed, the concept of generalization is quite vague and does not differentiate interpolation from extrapolation. Although our mathematical and operational grasp of interpolation is relatively good (Anastassiou, 1992; Brudnyi & Krugljak, 1991), the same does not hold true for extrapolation in the general case (see, however, Brezinski & Redivo Zaglia, 1991). Thus, effective general interpolators are available, but they are almost always poor extrapolators. Neural networks are no better than other interpolators in this respect, and their generalization capabilities depend on the location of the chosen generalization point with respect to the convex hull of the set of learning points, as we shall demonstrate below. It is therefore necessary to be able to determine the interpolation domain for a given learning set, that is, to estimate the neural network's domain of validity and to know the location of a given generalization point relative to that domain. Once this capability has been acquired, an educated system should be able to assess its own ability to treat a given problem, and potentially, to situate that problem in one domain or another if the system is competent in several domains (which is the case in natural cognitive systems). Needless to say, this capability is mandatory for assessing the reliability of a generalization response in real artificial intelligence applications. Moreover, in the case of human cognitive modelling, we can consider...
that an individual of normal intelligence is ordinarily capable of coming up with at least an approximate evaluation of his or her own ability to solve a given stimulus problem and to situate it in the proper cognitive domain.

From an operational standpoint, we can imagine some very simple solutions, such as calculating the distance between a point of generalization and the nearest known example, or taking the geometric mean of the distances between that point and all known examples, these two measures being equal to zero iff the point of generalization and some known point coincide. Unfortunately, the above solutions require that the entire learning set be available to the application, which in addition to being cumbersome and costly, does not actually take the system's generalization capabilities into account. To obtain a more global estimate of the domain of validity, one could, for example, determine the bounds (min and max) of the values of the learning set input variables and use the hyperrectangle, which is the product of these intervals, to approximate the domain. It is then perfectly simple to determine whether the point of generalization belongs to the hyperrectangle, which is described by a mere $2n$ values (where $n$ is the number of input variables). In practice, however, this hyperrectangle is often a gross overestimate and a very rough approximation of the interpolation domain.

Three relatively simple algorithmic solutions to this problem are proposed below. The first determines and uses the convex hull polytope of the set of learning points, representing the polytope solely by its vertices (in the input variable space). This polytope can be considered to delimit the interpolation support domain itself. The second solution is a good approximation of the first, and is performed by a recurrent neural network that is particularly easy to configure. The third solution should only be used when the polytope has too many vertices for the desired application. It consists of approximating the domain with the sphere circumscribing the learning set (and thus, the hull polytope), which always represents an overestimation of the interpolation domain. Two advantages of this third technique are that only the coordinates of the center and the radius of the sphere ($n + 1$ values) need be retained for the application, and the calculation of the distance from a point to the sphere is very simple.

2. FIRST ALGORITHM: INTERPOLATION SUPPORT POLYTOPE

2.1. Exteriority of a Point to a Convex Hull Polytope

Let $G = \{X_1, \ldots, X_m\}$ be a cluster of $m$ points in $\mathbb{R}^n$, and let $P(G)$ be the smallest convex subset of $\mathbb{R}^n$ containing $G$. $P(G)$ is delimited by the convex hull polytope of $G$. Hereafter, $n$ will be considered to be the number of input variables. The cluster $G$ can correspond to the learning set, or to any subset that contains all the vertices of the learning set's convex hull polytope.

Let $B$ be a vector of $m$ barycentric coordinates:

$$B = (b_1, \ldots, b_m) \in \mathbb{B}^m = \left\{ B; b_j \in \mathbb{R}_+, \sum_{j=1}^{m} b_j = 1 \right\}$$

Let $Y$ be a point in $\mathbb{R}^n$. Given the classical properties of barycenters, we have:

$$Y \in P(G) \iff \exists B \in \mathbb{B}^m: \left\| \left( \sum_{j=1}^{m} b_j X_j \right) - Y \right\| = 0.$$ 

The quantity defined by:

$$E(Y, G) = \min_{B \in \mathbb{B}^m} \left\| \left( \sum_{j=1}^{m} b_j X_j \right) - Y \right\|$$

is the euclidean distance between $Y$ and the point in $P(G)$ closest to $Y$. Let us call $E(Y, G)$ the exteriority of $Y$ to $P(G)$.

2.2. Minimization of $E$

Let:

$$b_j = a_j^2 \left/ \sum_{k=1}^{m} a_k^2 \right., \quad A = (a_1, \ldots, a_m) \in \mathbb{R}^m - \{0\}.$$ 

This gives us:

$$E^2(Y, G) = \min_{A \in \mathbb{R}^m - \{0\}} \left\| \left( \sum_{j=1}^{m} \frac{a_j^2}{\sum_{k=1}^{m} a_k^2} X_j \right) - Y \right\|^2.$$ 

It is easy to determine $A$ such that $A$ minimizes $E^2(Y, G)$, and consequently $E(Y, G)$, by using a standard gradient descent technique (see for example Zhigljavsky, 1991, pp. 21–23) with the following initial conditions: $a_j \neq 0, \forall j$. Indeed, depending on $Y$, zeroing certain $a_j$'s may lead to local maxima that should be avoided, and which a gradient descent technique would never land on unless initially required to do so. However, all minima (infinite in number) are global and provide the solution. Indeed, vector $A$ is not unique, and there are an infinite number of equivalent solutions that give the same measure $E(Y, G)$. Obviously, $E(Y, G)$ is unique because the inside point closest to $Y$ is itself unique due to the convexity of the polytope hull.

2.3. Extraction of the Polytope Vertices

It is useless to keep all points in cluster $G$ to define the convex hull polytope because only its vertices are necessary. Let $S(G)$ be the set of vertices of $P(G)$. Then $S(G)$ can simply be determined as follows: extract each point of $G$, one after the other, and for each point calculate its exteriority $E$ to the convex hull polytope of the remaining cluster of points. A given point is a vertex iff its exteriority is not equal to zero, in which case it
is reintegrated into the set. The set of points remaining at the end is $S(G)$.

**3. SECOND ALGORITHM: RECURRENT NEURAL NETWORK**

This algorithm is a recurrent neural network that approximates the convex hull polytope. It gives essentially the same results as the first algorithm, although the calculations are somewhat different, making it possible to implement standard neural systems. It has a recurrent four-layer architecture with one input layer and three processing layers (Figure 1). Hereafter the layers will be numbered from 0 to 3, with 0 as the input layer number. The activation level of the $i$th neuron in layer $k$ at time $t$ will be denoted $V_{ik}(t)$, and its response will be denoted $R_{ik}(t)$. The subscript $i$ will be omitted for the neuron in layer 3 only. The unit of time will be the synaptic transmission time, assumed to be constant and greater than zero. $x_j$ will be used to denote the $i$th coordinate of the $j$th point (among the $m$ points with $n$ coordinates) in the learning set or subset of that set used by the network. Likewise, $y_i$ denotes the $i$th coordinate of test point $Y$.

**Layer 0.** The input layer consists of $n$ cells, each of which receives one of the $n$ coordinates of test point $Y$. The response given by each cell is equal to the value it contains.

**Layer 1.** Layer 1 also consists of $n$ neurons, each of which corresponds to an input coordinate and receives a connection of synaptic weight 1 from the corresponding cell in layer 0. It also receives a connection from each of the neurons in layer 2. The synaptic weight of the connection linking the $j$th neuron in layer 2 to the $i$th neuron in layer 1 is $-x_j$. Layer 1 neurons have a linear response.

$$R_{1i}(t) = V_{1i}(t) = y_i - \sum_{j=1}^{m} x_j R_{2j}(t-1),$$

$t \geq 1, \quad R_{1i}(0) = 0$.

Function: the response of each cell in layer 1 is a difference that occurs in the exteriority gradient.

**Layer 2.** Layer 2 consists of $m$ neurons, each of which corresponds to one of the learning points used. Each neuron in layer 2 receives a connection from each neuron in layer 1, and the synaptic weight of the connection going from the $i$th neuron of layer 1 to the $j$th neuron of layer 2 is $u \cdot x_j$, where $u$ is a small positive constant. It also receives a self-connection with a synaptic weight equal to 1, and a connection weighing $-1$ from the sole neuron in layer 3. Layer 2 neurons have a semilinear response.

$$V_{2j}(t) = R_{2j}(t-1) + u \sum_{i=1}^{n} x_i R_{1i}(t-1) - R_3(t-1),$$

$$R_{2j}(t) = V_{2j}(t) \quad \text{if} \quad V_{2j}(t) \geq 0, \quad \text{else} \quad 0; \quad R_{2}(0) = 1/m.$$

Function: the response of layer 2 is an approximation of a vector of positive barycentric coordinates.

**Layer 3.** Layer 3 has only one neuron that receives a connection of synaptic weight 1 from each neuron in layer 2, and has a bias equal to $-1$. Its response is a signed square of $V$. This nonlinear smooth function was experimentally found to provide good convergence properties.

$$V_3(t) = \sum_{j=1}^{m} R_3(t-1) - 1$$

$$R_3(t) = V_3(t)|V_3(t)|, \quad R_3(0) = 0.$$ 

Function: This neuron keeps the sum of the layer-2 responses close to 1.

The constant $u$. The constant $u$ mentioned above acts like the step size in gradient algorithms that use a constant step size, and deciding what $u$ to choose poses comparable problems: a very small step size guarantees convergence but the process may be slower than necessary. Too large a step size can lead to divergent oscillations in the state of the process. The constant $u$ should be chosen in accordance with the scale of the variables. A simple and usually good approach is to choose a value somewhere in the vicinity of $10^{-3}/M$, where $M$ is the highest possible absolute value of the input coordinates.

**Convergence.** The recurrent process ends when the state of the network stabilizes, that is, when the variation in the state of each neuron from one instant to the next falls below a given cutoff point (which is analogous to the zero gradient test). When the test point $Y$ is inside the polytope (or belongs to it), the process ends with zeros in layers 1 and 3, and $m$ barycentric coordinates of $Y$ in layer 2. When the test point is outside as well as inside the polytope, a very good approximation of the exteriority $E(Y, G)$ can be obtained by:

$$\forall j: \quad b_j = R_{2j}(V_3 + 1),$$

and

$$E(Y, G) \approx \left\| \sum_{j=1}^{m} b_j x_j - Y \right\| .$$

**Extraction of the vertices of the polytope hull.** The procedure is similar to that employed in the preceding algorithm. The neurons in layer 2 are selected.

**4. THIRD ALGORITHM: CIRCUMSCRIBED SPHERE**

When the polytope has a large number of vertices, a more economical technique should be used. The most efficient approximation method known to the author in this case is the polytope's circumscribed sphere, which is uniquely defined by its center $X$ and its radius $r$, that is, by $n + 1$ real numbers. Moreover, the exteriority of any point $Y$ to this sphere is easy to calculate.
It is equal to $\| Y - X \| - r$ if this quantity is positive, and to 0 if it is not.

4.1. Calculation of the Circumscribed Sphere

The sphere that circumscribes the convex polytope hull of a cluster of $m$ points in $\mathbb{R}^n$ can be obtained by solving the following minimization problem:

$$\min_{X \in \mathbb{R}^n} \left( \sup_{1 \leq j \leq m} \| X_j - X \|^2 \right).$$

This gives the center $X$ of the circumscribed sphere, as well as its radius:

$$r = \left( \sup_{1 \leq j \leq m} \| X_j - X \|^2 \right)^{1/2}$$

The usual sup operator over a discrete set defines a $C^0$ continuous function (i.e., non-smooth) of $X$, which is not very suitable for minimizing with simple algorithms of the gradient descent type. However, the function is simple and the application of a sophisticated global optimization algorithm is not justified. Different tricks can be used, and the one proposed here consists of redefining the sup operation by a smooth convergent approximation. Let:

$$M_s(v_1, \ldots, v_m) = \frac{1}{s} \ln \left[ \sum_{j=1}^{m} \exp(s v_j) \right].$$

It is easy to prove that $\lim_{s \to \infty} M_s = \sup_{1 \leq j \leq m} v_j$, and $\lim_{s \to -\infty} M_s = \inf_{1 \leq j \leq m} v_j$.

By choosing a sufficiently large $s$, this smooth approximation can be substituted for the sup operator in the above expression for the circumscribed sphere. $X$ (and consequently $r$) can then be quickly calculated by a standard gradient descent algorithm. It is generally a good idea at the beginning of the search to set the initial value of $X$ at the cluster's center of gravity because this not only slightly reduces computation time, but above all, allows for a larger $s$ (and thus, greater precision) without risking overflow in the calculation of the exponential functions.

Limitations for use. The circumscribed sphere always overestimates the interpolation support domain, and is only a usable approximation when the shape of the polytope resembles that of a sphere. This is frequently the case when the input variables are independent and have the same range of variation. If so, the circumscribed sphere provides some very good information because neuron interpolators generally extrapolate quite effectively in the neighborhood of the polytope. Otherwise, it would be preferable to use one of the first two algorithms, retaining as vertices the points with the highest exteriority to the polytope hull of the remaining points, thereby reducing the number of vertices but underestimating the interpolation domain. Note also that there is another more conventional approach that consists of estimating the domain by a hyperellipsoid based on the eigen axes of the cluster of learning points in the input variable space. The approach is a more general one, but is also more complex than the circumscribed sphere technique.

5. RELATION BETWEEN EXTERIORITY AND GENERALIZATION ERROR

It has been experimentally observed, in a large variety of problems, that the mean and variance of the generalization error of neural smoothers systematically increases with the exteriority of the generalization points considered. This relation accounts statistically for a large part of the generalization error. The generalization error also depends more locally on how far the generalization point is from a learning point, and of course, on the specific characteristics of the objective function, the learning sample, and the interpolator itself. A value of zero for the exteriority of a generalization point to the interpolation polytope does not guarantee an exact response by the interpolator, but the potential error and its range of variation are smaller than for larger exteriority values.

5.1. Computational Experiment

To illustrate, let us take as an example the approximation of random three-variable polynomials performed by neural networks generated by a convergent
NN Validity Domain

The objective functions have the general form:

\[ f(x, y, z) = \sum_{i=0}^{3} \sum_{j=0}^{3} \sum_{k=0}^{3} w_{ijk} x^i y^j z^k, \]

with \( w_{ijk} \) taken at random from a uniform distribution on \([-1, 1]\) for each polynomial.

Ten such polynomials were generated for the experiment. For each polynomial, 64 points taken at random from a uniform distribution in the cube \([-1, 1]^3\) were used as the learning set, with their corresponding function values. Sixty-four points taken at random from a uniform distribution in the cube \([-2, 2]^3\) were used as the generalization test points. For each problem, the learning set was input into the neural network generator algorithm. The convergence properties of this algorithm are such that one can predefine a residual error level (filtering) for the examples at the end of the learning phase. Given that the data were exact, we chose a low filtering level with an absolute error tolerance of less than \(10^{-3}\) per point. The generator algorithm produced neural networks whose complexity (number of hidden units and connections) depended on the problem considered. For each learning set, the convex hull polytope and the circumscribed sphere were computed using the above algorithms (for computing the circumscribed spheres, \( s \) was set at 1200). For each generalization test point, the neural network generalization error, the exteriority to the convex hull polytope, and the exteriority to the circumscribed sphere were computed. The relation between the absolute errors and the two types of exteriority was tested using linear correlation coefficients (usual Bravais-Pearson's \( r \) statistic). The \( r \) statistics here have 62 degrees of freedom. The theoretical \( r \) distribution states that \( \text{prob}\{ | r_{20} | > 0.40 \} < 0.001 \), so \( r > 0.40 \) corresponds to the usual threshold for a very high level of statistical significance. The linear correlation test is very severe in the present case because the \( r \) statistic is an increasing function of the covariance, but a decreasing function of the variance, and as a general rule, the variance of generalization errors increases with the exteriority. For each of the 10 problems, Table 1 gives the number of hidden units generated, the number of vertices of the polytope, the absolute generalization error range inside the polytope, the absolute generalization error range outside the polytope, the linear correlation between the absolute generalization error and the exteriority to the polytope (\( r_p \)), and the linear correlation between the absolute generalization error and the exteriority to the sphere (\( r_s \)).

As one can see in Table 1, all the correlations between the two types of exteriority and the absolute generalization error were positive and very highly significant. Correlation coefficients can be compared using a \( z \)-transformation. None of the differences between \( r_p \) and \( r_s \) were statistically significant, and globally, the 20 correlation coefficients were not significantly different from each other \((x^2 = 9.19, df^2 = 19, \alpha = 0.97)\). The global estimation of the correlation gave \( \rho^* = 0.62 \), and \( \text{prob}\{ 0.56 \leq \rho \leq 0.67 \} = 0.999 \). As one can see, even the lower confidence limit of the correlation between the exteriority and the absolute generalization error remains very highly significant \((\alpha = 1.5 \times 10^{-6})\). Note also the large difference between the absolute generalization error ranges inside and outside the polytope. The exteriority to the sphere appeared to predict the generalization error at least as well as the exteriority to the polytope did. Note, however, that this would not be the case if at least two of the input variables were not independent or had different variation ranges.

**6. CONCLUSION**

The first two algorithms presented here determine and use the convex hull polytope of a cluster of points in \( \mathbb{R}^n \) that corresponds to the interpolation support domain of a function sampled over the domain. The first algorithm is in a conventional form, whereas the second is in a neuron-like form that allows for a different type of implementation. The third algorithm approximates the interpolation domain using the circumscribed

<table>
<thead>
<tr>
<th>Hidden Units</th>
<th>Polytope Vertices</th>
<th>Absolute Error Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inside</td>
<td>Outside</td>
<td>( r_p )</td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>0.063-0.208</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>0.002-0.068</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>0.012-0.103</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>0.027-0.066</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>0.033-0.076</td>
</tr>
<tr>
<td>6</td>
<td>23</td>
<td>0.017-0.134</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>0.080-0.169</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>0.010-0.183</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
<td>0.065-0.346</td>
</tr>
<tr>
<td>10</td>
<td>22</td>
<td>0.028-0.087</td>
</tr>
</tbody>
</table>
sphere, which is economical and sufficient in certain cases. Experimental studies have shown that the generalization error tends to rise and become increasingly unpredictable as the exteriority of the test point to the interpolation domain increases. It is obvious that other factors influence the error level. For example, the distance to the nearest control point is a well-known factor, and it is in fact correlated with the exteriority value studied here. The advantage of the approach we are proposing is that it reduces the amount of coding: the entire learning set does not have to be entered with the application in order to effectively estimate the reliability of the generalization responses. Finally, note that the proposed methods estimate the reliability of generalizations with respect to a given learning set, but do not test the validity of the learning set with respect to the objective function. So they cannot guard against the consequences of a bad sampling of the objective function in the learning set, even if the neural learning algorithm used is highly convergent. For example, if some of the points where the objective function undergoes critical changes (e.g., local extrema, curvature extrema, etc.) are missing from the learning set, then there will probably be a large generalization error on the neighborhood of these points, even if they belong to the interpolation domain. When the learning set is random, a value of zero for the exteriority of a generalization point does not guarantee a correct answer by the neural network, but a nonnegligible exteriority indicates that the answer is probably wrong.

REFERENCES