Towards Faster, Smoother, and More Compact Fuzzy Approximation, with an Application to Non-Destructive Evaluation of Space Shuttle's Structural Integrity

Yeung Yam
Department of Mechanical & Automation Engineering
The Chinese University of Hong Kong
Shatin, NT, Hong Kong, China
yyam@mae.cuhk.edu.hk

Roberto Osegueda and Vladik Kreinovich
Future Aerospace Science and Technology Program (FAST)
Center for Structural Integrity of Aerospace Systems
University of Texas at El Paso
El Paso, TX 79968, USA
{osegueda,vladik}@utep.edu

Abstract

It is known that fuzzy systems are universal approximators, i.e., any input-output system can be approximated, within any given accuracy, by a system described by fuzzy rules. Fuzzy rules work well in many practical applications. However, in some applications, the existing fuzzy rule approximation techniques are not sufficient:

First, in many practical problems (e.g., in many control applications), derivatives of the approximated function are very important, and so, we want not only the approximating function to be close to the approximated one, but we also want their derivatives to be close; however, standard fuzzy approximation techniques do not guarantee the accuracy of approximating a derivative.

Second, to get the desired approximation accuracy, we sometimes need unrealistically many rules.

In this talk, we show how both problems can be solved.

1. First Problem: In Many Practical Situations, It Is Desirable to Have a Smooth Extrapolation

1.1. Extrapolation is important

In many practical problems, we know the values of a function $f(x)$ at several points $x^{(1)}, \ldots, x^{(k)}$, and we are interested in knowing the values of the function $f(x)$ for all x, i.e., we want to *extrapolate* the function $f(x)$. For example:

In *measurement*, we measure the values of some distributed characteristic at places where the sensors are located, and we want to reconstruct the values of this characteristic at all points.

In *control*, we may have recorded the control values $f(x^{(1)}), ..., f(x^{(k)})$ used by an experienced operator for different inputs $x^{(1)}, \ldots, x^{(k)}$, and we want to design an automated controller $f(x)$ that would apply exactly these control values for these inputs.

In *image processing*, we observe the intensity at several points, and we would like to reconstruct the entire image (i.e., the intensity at other points) based on these known values. This is necessary, e.g., if we want to *rotate* an image, because the image is normally stored by its values on a rectangular grid, and when we rotate,

the pixels from the grid no longer coincide with the grid values, so we need to extrapolate.

There are many successful extrapolation techniques:

- we can use numerical methods like *spline* (piece-wise polynomial) extrapolation;
- we can train a *neural network* on the patterns $(x^{(1)}, f(x^{(1)})), \ldots, (x^{(k)}, f(x^{(k)}))$, and then use, for an arbitrary input x, the result $f(x)$ of the network's training as the desired output;
- we can use *fuzzy* extrapolation techniques, namely, we can form a rule-base with the rules "if x is close to $x^{(i)}$, then $f(x)$ should be close to $f(x^{(i)})$" $(1 \le i \le k)$, and apply the general fuzzy modeling methodology to transform these rules into the precise control strategy (see, e.g., [5]).

1.2. In many practical problems, we want to preserve smoothness

The above extrapolation techniques provide a good approximation to the actual (unknown) values of the desired function $f(x)$ in the sense that for every input x, the extrapolated value $\widetilde{f}(x)$ is usually close to the value of the function $f(x)$. In many practical applications, however, it is not enough that the *values* of the desired functions be approximated precisely; it is also desirable that the values of its *derivatives* be reproduced accurately. For example:

In *nondestructive testing* of structural integrity, we send an ultrasonic signal to the tested system, and measure the resulting vibration at different points. Our goal is to detect the points where the cracks or other possible faults are. It is known that faults correspond to large values of stress, and stress can be evaluated as a curvature (second derivative-related characteristic) of the observed vibration amplitude. So, for applications to nondestructive testing, we need to be able to approximation second derivatives correctly.

In *control* applications, we often want control to be *smooth* (e.g., in control related to transportation of human beings or delicate goods; in docking a spaceship to a space station, etc). Therefore, when we start with the recorded experience of an expert controller (who knows how to control it smoothly), and we extrapolate the control from the operator-given values $f(x^{(k)})$, we not only want the resulting automatic controller to reproduce the *values* of the resulting control function $f(x)$ accurately, we also want this controller to accurately reproduce the *derivatives* of the expert's control function.

In *image processing*, it is also often very important to reproduce smoothness accurately.

For example, one of the important image processing problems is finding text in a pixel-by-pixel image. This problem is extremely important for *security*, when to detect potential threats, it is desirable to look up millions of webpages to see if any of them contain potentially dangerous words and phrases. It is easy to use web search tools to detect these words as text, but new methods are needed to detect these words in web-placed images. A natural way to detect a text in an image is to use the fact that a text is a *discontinuous* (or at least non-smooth) part of the image. Since an image can contain a text which is neither horizontal nor vertical, a reasonable way to text finding includes a rotation of the text. Therefore, when we make a rotation-related extrapolation, we would like to keep non-smooth parts non-smooth, and smooth parts smooth (to preserve the ability to detect the given image).

In analyzing *satellite photos*, there is also a need for rotations: images of nearby areas are often obtained from slightly different angles, so we must rotate them before we can combine ("mosaic") then into a single large-area picture. Many useful geophysical features like faults (or geographical features like roads) represent non-smoothnesses in an image; therefore, it is important that the related rotation preserve smoothness.

1.3. Existing techniques do not always preserve smoothness: a problem

Not all existing extrapolation techniques preserve smoothness.

For example, the simplest *spline* extrapolation – piece-wise linear extrapolation of a function – preserves its values, and describes the first derivatives of this function more or less accurately. However, the resulting first derivative is piece-wise constant; therefore, the second derivative of the approximating function consists of 0's separated by infinite values at $x^{(i)}$. As a result, this extrapolation is useless for nondestructive testing, where we need second derivatives.

Standard *fuzzy* extrapolation is based on a very localized membership function describing "close"; as a result, we get the same problem with second derivatives.

Since some extrapolation techniques do approximate derivatives well (e.g., some neural networks), one possible solution may be to abandon the existing techniques and use only those which preserve derivatives. However, we may not want to simply abandon these techniques, because they have many advantage; e.g., fuzzy rules have two major advantages:

- first, fuzzy rules (in contrast to, say, PDE's or neural networks) are intuitively clear;
- second, fuzzy rule representation is naturally parallelizable, which is important if we want to save computation time.

It is due to these advantages that fuzzy rules work well in many practical applications. So, instead of simply abandoning these techniques, we would like to *modify* the existing techniques so that they will approximate derivatives as well. In this paper, we describe the necessary modifications.

2. How to Make Approximation Smooth: Practical Technique

We will describe this technique by presenting cases of increasing complexity until we get the most general situation.

2.1. Case 1: function of one variable, first-order derivative desired

In this simple case, we know the values $f_i = f(x^{(i)})$ of an unknown function $f(x)$ at points $x^{(1)}, \ldots, x^{(k)}$; without losing generality, we can assume that $x^{(1)} < x^{(2)} < \ldots < x^{(k)}$. Our goal is to provide an extrapolation $f(x)$ which would approximate *both* the values of the function $f(x)$ *and* of its derivative $f'(x)$. We know the desired values f_i of the function $f(x)$; we can also use standard numerical differentiation formulas to estimate the values of its derivatives, e.g., as $f'(x^{(i)}) \approx d_i = (f_{i+1} - f_{i-1})/(x^{(i+1)} - x^{(i-1)})$. (This two-sided *symmetric* derivative is preferable because its error is much smaller than for a more standard one-sided asymmetric expression $(f_{i+1} - f_i)/(x^{(i+1)} - x^{(i)})$; in border points, we have no other choice but to use a one-sided formula.)

In other words, the problem is to find a function $f(x)$ which satisfies k conditions $f(x^{(i)}) \approx f_i$ and k conditions $f'(x^{(i)}) \approx d_i$. A natural way to solve this problem is to first satisfy half of these conditions, and then modify the solution to satisfy the other half. We have already mentioned that if we start by applying one of the known extrapolation techniques to satisfy the conditions $f(x^{(i)}) \approx f_i$, then we may get completely wrong values of the derivatives. So, the only remaining possibility is to first extrapolate the derivatives. Thus, on the *first stage* of the proposed algorithm, we compute the values d_i, and use the chosen technique to find a function $\widetilde{d}(x)$ for which $\widetilde{d}(x^{(i)}) = d_i$.

This function $\widetilde{d}(x)$ is an approximation to the derivative $f'(x)$; therefore, we can reconstruct the desired function $f(x)$ as $I(x) + C$, where $I(x) = \int_0^x \widetilde{d}(y)\,dy$, and the integration constant C can be determined, e.g., by applying the Least Squares Method to k conditions $f(x^{(i)}) = I(x^{(i)}) + C \approx f_i$: in other words, $C = (1/k) \cdot \sum_i (f_i - I(x^{(i)}))$. So, the *second stage* of the algorithm consists of computing the integral $I(x)$ (by numerical integration), finding the value C from the above formula, and then computing $\widetilde{f}(x) = I(x) + C$.

Since numerical differentiation only gives *approximate* values of the derivatives $d_i \approx f'(x^{(i)})$, as a result of this two-stage algorithm, we get a function $\widetilde{f}(x)$ for which $\widetilde{f}(x^{(i)}) \approx f_i$. In many practical applications, the values f_i come from measurements and are, therefore, themselves imprecise; so, it is quite sufficient to have $\widetilde{f}(x^{(i)}) \approx f_i$.

In some cases, however, the measurements which lead to f_i are very precise, so we would like to get the exact equality $f(x^{(i)}) = f_i$ (or at least to decrease the approximation error $|f(x^{(i)}) - f_i|$). In such cases, since we already know that $f(x) \approx \widetilde{f}(x)$, we represent the desired function $f(x)$ as $\widetilde{f}(x) + \Delta_1(x)$, with $\Delta_1(x^{(i)}) = f(x^{(i)}) - \widetilde{f}(x^{(i)}) = f_i - \widetilde{f}(x^{(i)})$, and repeat the above two-stage procedure for this new function $\Delta_1(x)$. As a result, we get an approximation $\widetilde{\Delta}_1(x) \approx \Delta_1(x)$, with $|\widetilde{\Delta}_1(x^{(i)}) - \Delta_1(x^{(i)})| \ll |\Delta_1(x^{(i)})|$, and therefore, a better approximation $\widetilde{f}_1(x) = \widetilde{f}(x) + \widetilde{\Delta}_1(x)$ for $f(x)$. If we are still not satisfied with the error level of this approximation, we can form a new difference $\Delta_2(x) = f(x) - \widetilde{f}_1(x)$, and repeat the same error-decreasing procedure, etc., until the approximation error is small enough.

2.2. Case 2: function of one variable, higher-order derivatives desired

We already know how to approximate a function $f(x)$ of one variable x together with its first derivative $f'(x)$. We can use the same idea to approximate a function together with any number of derivatives. Indeed, let us show, e.g., how to approximate a function $f(x)$ and its two derivatives $f'(x)$ and $f''(x)$. By using numerical differentiation, we get the values d_k of the first derivative $f'(x)$. To these values, we apply the above two-stage algorithm and get an approximation $\widetilde{d}(x)$ for the function $f'(x)$ *and* for its first derivative $(f')'(x) = f''(x)$. From this approximation, we can reconstruct $f(x)$ as $\widetilde{f}(x) = I(x) + C$, where $I(x) = \int_0^x \widetilde{d}(y)\,dy + C$, and a constant C can be determined similarly to Case 1. The resulting algorithm approximates a function together with the values of its first two derivatives. (Similarly to Case 1, if we are not satisfied with the approximation error,

we can repeat the same procedure for the difference $\Delta_1(x) = f(x) - \widetilde{f}(x)$, and take $\widetilde{f}_1(x) = \widetilde{f}(x) + \widetilde{\Delta}_1(x)$ as the next approximation to $f(x)$.)

A similar recursive procedure helps to approximate a function together with any given number of derivatives. Indeed, suppose that we already have a procedure $\mathcal{P}_s$ which approximates a function $f(x)$ together with its s derivatives $f'(x), f''(x), \ldots$, and we want to approximate a function together with its $s+1$ derivatives. By using numerical differentiation, we get the values d_i of the first derivative $f'(x)$ of the desired function. To these values, we apply the algorithm $\mathcal{P}_s$ and get an approximation $\widetilde{d}(x)$ for the function $f'(x)$ *and* for its s first derivatives. From this approximation, we can reconstruct $f(x)$ as $\widetilde{f}(x) = I(x) + C$, where $I(x) = \int_0^x \widetilde{d}(y)\,\mathrm{d}y + C$, where a constant C can be determined similarly to Case 1. The resulting algorithm approximates a function together with its first derivative, and with the first s derivatives of this derivative, i.e., it reconstructs a function together with its $s+1$ derivatives.

2.3. General case: function of several variables

In the following text, we follow a convenient notation (widely used in physics) of denoting a partial derivative of a function f with respect to a variable x_i by $f_{,i}$, and, correspondingly, a partial derivative with respect to x_i and x_j by $f_{,ij}$. A general derivative has the form $f_{,\mathcal{D}}$, where $\mathcal{D} = i_1 \ldots i_s$ is a sequence of indices (i.e., the form $f_{,i_1 \ldots i_s}$).

If we want to reconstruct a function $f(x_1, x_2, \ldots, x_n)$ of several variables together with its first derivative $f_{,1}$, then we can apply the same two-stage procedure as for a function of one variable, with the only complication that after computing the numerical values d_i of the first derivative and extrapolating these values into a function $d(x_1, x_2, \ldots)$, we have $\widetilde{f}(x_1, x_2, \ldots) = I(x_1, x_2, \ldots) + C(x_2, \ldots)$, where $I(x_1, x_2, \ldots) = \int_0^{x_1} d(y, x_2, \ldots)\,\mathrm{d}y$, and the integration constant $C(x_2, \ldots)$ may depend on the variables $x_2, \ldots$ We know that for given inputs $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$, we must have $C(x_2^{(i)}, \ldots, x_n^{(i)}) \approx f_i - I(x^{(i)})$; therefore, to find the function $C(x_2, \ldots)$, we need a *third stage*, on which we apply the same original extrapolation procedure to these values to get the function $C(x_1, \ldots)$.

If we want to reconstruct a function $f(x_1, \ldots, x_n)$ together with several derivatives (e.g., with its two first derivatives $f_{,1}$ and $f_{,2}$), then we first reconstruct the lowest derivative from which both $f_{,1}$ and $f_{,2}$ can be obtained by integration (in the above example, $f_{,12}$), and then find f by several consequent integrations.

2.4. Practical application

As a case study, we applied the new methods to the problem of non-destructive evaluation of structural integrity of Space Shuttle's vertical stabilizer. To prove the applicability of our method, we applied this techniques to measurement results for pieces with known fault locations. Our method detected all the faults in >70% of the cases, much larger proportion than with any previously known techniques (for details, see [1, 6, 7, 8]).

2.5. Theoretical Justification

It is known that fuzzy systems are universal approximators, i.e., any input-output system can be approximated, within any given accuracy, by a system described by fuzzy rules (see, e.g., a survey [5], pp. 135–195, and references therein). To guarantee that a smooth approximation is always possible, let us prove that, even if we understand accuracy as closeness of both the approximation and its derivatives, fuzzy systems are still universal approximators.

In this result, we follow [9] and consider fuzzy systems in which "and" is represented by an algebraic product, aggregation is represented by sum, membership functions for the input are Gaussian, and outputs are crisp. For such systems, the input-output function is given by a formula

$$g(x_1, \ldots, x_n) = N(x_1, \ldots, x_n)/D(x_1, \ldots, x_n), \quad \text{(1a)}$$

where

$$N(x_1, \ldots, x_n) = \sum_{i=1}^{m} w_i \cdot \mu_{i,1}(x_1) \cdot \ldots \cdot \mu_{i,n}(x_n), \quad \text{(1b)}$$

$$D(x_1, \ldots, x_n) = \sum_{i=1}^{m} \mu_{i,1}(x_1) \cdot \ldots \cdot \mu_{i,n}(x_n), \quad \text{(1c)}$$

w_i are real numbers, and $\mu_{i,j}(x)$ are Gaussian functions.

Let s be an integer, let $\varepsilon > 0$ be a real numbers. We say that a function $g(x_1, \ldots, x_n)$ *approximates* a function $f(x_1, \ldots, x_n)$ and its derivatives of order $\le s$ with accuracy ε if for all $x_i \in [-\Delta, \Delta]$,

$$|f(x_1, \ldots, x_n) - g(x_1, \ldots, x_n)| \le \varepsilon,$$

and for every derivative $\mathcal{D}$ of order $\le s$,

$$|f_{,\mathcal{D}}(x_1, \ldots, x_n) - g_{,\mathcal{D}}(x_1, \ldots, x_n)| \le \varepsilon.$$

Theorem. *Let s and n be integers, let $\Delta > 0$ and $\varepsilon > 0$ be real numbers, and let $f(x_1, \ldots, x_n)$ be an s-times differentiable function on $[-\Delta, \Delta]^n$. Then, there exists a function $g(x_1, \ldots, x_n)$ of type* (1a)–(1c) *which approximates $f(x_1, \ldots, x_n)$ and its derivatives of order $\leq s$ with accuracy ε.*

The proof of this result is given in [4].

3. Second Problem: It Is Often Desirable to Have Fewer Fuzzy Rules

Most current applications of fuzzy control deal with reasonably simple systems, where a small number of control rules is sufficient; straightforward application of the same methodology to more complicated systems leads, sometimes, to unrealistically many rules. This problem (and methods of handling it) has been described, e.g., in [2, 3].

We will show how, for a reasonable class of fuzzy controllers, we can decrease the number of rules. Let's consider a system with inputs x_1, x_2 and control rules "if $A_{1,i}(x_1)$ and $A_{2,j}(x_2)$ then $u = u_{i,j}$", where $A_{\ell,j}$ are fuzzy properties, and $u_{i,j}$ are given values. We assume that for $\ell = 1, 2$, the corresponding properties $A_{\ell,i}$ form a *fuzzy partition*, i.e., for every x_ℓ, $\sum_i A_{\ell,i}(x_\ell) = 1$. If we use $a \cdot b$ as a t-norm, addition for combining rules, and center-of-gravity defuzzification, the resulting control is $u(x_1, x_2) = \sum_{i,j} u_{i,j} \cdot A_{1,i}(x_1) \cdot A_{2,j}(x_2)$.

The fewer non-zero coefficients $u_{i,j}$, the fewer rules we need. So, to decrease the number of rules, we represent the corresponding bilinear form $F = \sum_{i,j} u_{i,j} \cdot y_i \cdot z_j$ as $F = \sum_{p=1}^{P} u'_p \cdot Y_p \cdot Z_p$, where Y_p are linear combinations of y_i, Z_p are linear combinations of z_j ($Y_p = \sum_i c^{(1)}_{p,i} \cdot y_i$, $Z_p = \sum_j c^{(2)}_{p,j} \cdot z_j$), and the number P is the smallest possible. Then, we can describe the fuzzy controller as

$$u(x_1, x_2) = \sum_p u'_p \cdot A'_{1,p}(x_1) \cdot A'_{2,p}(x_2),$$

where $A'_{\ell,p}(x_\ell) = \sum_i c^{(\ell)}_{p,i} \cdot A_{\ell,i}(x_\ell)$ are the corresponding linear combinations. Thus, P rules are sufficient.

If the matrix $u_{i,j}$ is symmetric, then the desired representation of the bilinear form corresponds to eigenvalues u'_p and eigenvectors $Y_p = Z_p$; in general, the desired representation is known as a Singular Value Decomposition (SVD). A similar reduction can be achieved if we have three or more inputs x_i. Practical examples (see, e.g., [10]) show that this reduction can indeed be drastic.

Comment. It is worth mentioning that by formally applying the SVD techniques, we sometimes end up with *negative* (or > 1) values of membership functions $A'_{\ell,p}$.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants No. DUE-9750858 and CDA-9522207, by the United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518, by the National Security Agency under Grant No. MDA904-98-1-0564, and by Hong Kong RGC Earmarked Grant CUHK519/95E.

References

[1] G. Andre, *Comparison of Vibrational Damage Detection Methods in an Aerospace Vertical Stabilizer Structure*, Master Thesis, The University of Texas at El Paso, Civil Engineering Department, May 1999.

[2] L. T. Kóczy and D. Tikk, "Approximation in rule bases", *Proc. IPMU'96*, Granada, Spain, 1996, pp. 489–494.

[3] B. Kosko, "Optimal fuzzy rules cover extrema", *Proceedings of the World Congress on Neural Networks WCNN'94*, 1994.

[4] V. Kreinovich, H. T. Nguyen, and Y. Yam, "Fuzzy Systems Are Universal Approximators for a Smooth Function And Its Derivatives", *The Chinese University of Hong Kong*, Technical Report CUHK-MAE-99-002, January 1999.

[5] H. T. Nguyen and M. Sugeno (eds.), *Fuzzy Systems: Modeling and Control*, Kluwer, Boston, MA, 1998.

[6] R. A. Osegueda, A. Revilla, L. Pereyra, and O. Moguel, "Fusion of modal strain energy differences for localization of damage", In: A. K. Mal (ed.), *Nondestructive Evaluation of Aging Aircraft, Airports, and Aerospace Hardware III*, Proceedings of SPIE, Vol. 3586, Paper 3586–28.

[7] L. R. Pereyra, R. A. Osegueda, C. Carrasco, and C. Ferregut, "Damage detection in a stiffened plate using modal strain energy differences", Ibid, Paper 3586–29.

[8] N. S. Stubbs, T. Broom, and R. A. Osegueda, "Nondestructive construction error detection in large space structures", *AIAA ADM Issues of the International Space Station*, AIAA, Williamsburg, Virginia, April 1998, pp. 47–55.

[9] L.-X. Wang, "Fuzzy Systems Are Universal Approximators", *Proc. FUZZ'IEEE*, San Diego, CA, 1992, pp. 1163–1170.

[10] Y. Yam, "Fuzzy approximation via grid point sampling and SVD", *IEEE Transactions on Systems, Man, and Cybernetics*, 1997, Vol. 27, No. 6, pp. 933–951.