Neural Networks, Qualitative-Fuzzy Logic and Granular Adaptive Systems

T. Y. Lin

Department of Mathematics and Computer Science
San Jose State University,
San Jose, CA 95192
tylin@cs.sjsu.edu

Abstract—Though traditional neural networks and fuzzy logic are powerful universal approximators, however without some refinements, they may not, in general, be good approximators for adaptive systems. By extending fuzzy sets to qualitative fuzzy sets, fuzzy logic may become universal approximators for adaptive systems. Similar considerations can be extended to neural networks.

Keywords: granular, neural network, qualitative fuzzy logic, adaptive system.
Glossary
1. Introduction
2. A Changing “PHYSICS”
   2.1 Shrinking Earth
   2.2 Increasing Gravitational Force
3. NN in Different Instances
   3.1 The NN of Some Experiments
      3.1.1 Initialization of Parameters
      3.1.2 The Case < P = 0.6 >
      3.1.3 The Case < P = 1.2 >
      3.1.4 The Case < P = 5.0 >
      3.1.5 The Case < P = 10 >
      3.1.6 The Case < P = 20 >
      3.1.7 The Case < P = 267 >
   3.2 Discussions
4. Neural Network
   4.1 Universal Approximations
      4.1.1 Universal Approximation Theorem (UAT)
      4.1.2 Practical Realization of UAT
         4.1.2.1 Initialization
         4.1.2.2 Fine Tuning
   4.2 NN-Learning and Adapting
      4.2.1 Training Data and “Spotted” Curve
      4.2.2 Patterns and “Continuous” Curve
4.3 Adaptive Dynamic Systems
4.3.1 Training Data EM(p)
4.3.2 The Pattern NN(p)
4.3.3 Key Questions
4.3.4 The Dynamic System of Dynamic Systems
4.3.5 Granular Neural Networks
5. Qualitative Fuzzy Logic
5.1 Traditional FL
5.2 FL on Qualitative Fuzzy Sets
6. Conclusions
Appendix 1
References
GLOSSARY

**Neural Network**: A type of artificial intelligence that attempts to imitate the way a human brain works. Rather than using a digital model, in which all computations manipulate zeros and ones, a neural network works by creating connections between processing elements, the computer equivalent of neurons.

**Fuzzy Logic**: A type of logic that recognizes more than simple true and false values. With fuzzy logic, propositions can be represented with degrees of truthfulness and falsehood. For example, the statement, today is sunny, might be 100% true if there are no clouds, 80% true if there are a few clouds, 50% true if it's hazy and 0% true if it rains all day.

**Universal Approximators**: Universal approximators are the one which have the ability to approximate any function to an arbitrary degree of accuracy. Neural nets are universal approximators.

**Fuzzy Sets**: Fuzzy sets are sets whose elements have degrees of membership. Fuzzy sets have an extension of the classical notion of set. In classical set theory, the membership of elements in a set is assessed in binary terms according to a bivalent condition — an element either belongs or does not belong to the set. By contrast, fuzzy set theory permits the gradual assessment of the membership of elements in a set; this is described with the aid of a membership function valued in the real unit interval.

**Dynamic System**: The dynamical system concept is a mathematical formalization for any fixed "rule" which describes the time dependence of a point's position in its ambient space. Examples include the mathematical models that describe the swinging of a clock pendulum, the flow of water in a pipe, and the number of fish each spring in a lake.

**Defuzzication**: Defuzzication is a very important part of the Fuzzy Adaptive Intervention Plan as it is the procedure that leads to decisions about the configuration of space and the architectural properties of different objects and spaces. Our speculation is that we need to develop another Neuro-Fuzzy controller that will take up the part of weighing and tuning different rules and proposals regarding the interpretation of the fuzzy output which results from the inference mechanism.

**Approximation theorem**: An approximation is an inexact representation of something that is still close enough to be useful. Although approximation is most often applied to numbers, it is also frequently applied to such things as mathematical functions, shapes, and physical laws. Approximations may be used because incomplete information prevents use of exact representations.
1. INTRODUCTION

Neural networks (NN) and fuzzy logic (FL) are well known for their learning capability. The mathematics behind it is the universal approximation theorems; e.g., [9] and [2]. Roughly, the theorem asserts that given a real valued function with reasonable properties, then there is a neural network or fuzzy logic that approximates the given function within a given error. This theorem provides the foundation for the common practices of "training neural networks," or “tuning.” Many practitioners have somewhat confused view about learning and adaptive. In this paper, we reason that they are different.

As Kosko observed that the learning or adaptation are merely changing of parameters [2]. However, a basic fact that is often not well recognized is that there are limitations on parameter changing. Intuitively once a parameter reaches a zero; it disappears from the models or equations. This fact limits the capability of adaptation and differentiates the learning from adaptation. Adaptation is not re-learning.

Before we go to the details, let us outline the general idea: Suppose we have a function $f$ (only numerically known) that models a physical phenomenon. The function $f$, of course, changes if physics changes. Based on these numerical data, a neural network or fuzzy logic system, denoted by $NN_f$ and $FL_f$, can be trained to approximate $f$. Then an important question is:

If physics changes, that is, the function $f$ is deformed into a new function, say $g$,
Could $NN_f$ or $FL_f$ be transformed into $NN_g$ or $FL_g$ respectively by merely adjusting the values of existing parameters?

Previously we observed that the universal approximators (for NN) do not guarantee, by merely adjusting its weights, the “same” NN could approximate $g$ [8]. In fact later, we even gave explicit examples for it [5]. In this paper, we propose to solve them by granular neural networks and qualitative fuzzy sets.

For simplicity, in this paper, we restrict the discussions to radial basis neural networks (similar arguments should hold for other neural networks) and fuzzy logic formulated in [2].

2. A CHANGING “PHYSICS”

The purpose of this section is to provide a mathematical model of changing “physics” to understand the power of NN and FL in building adaptive systems.

2.1 Shrinking Earth

Let us picture the following situation: The Earth, for some reasons, is loosing its radius very slowly without losing its mass. To be specific, the Earth is compacting itself in the following fashion:

1. The shrinking is very minute and are unnoticeable in short duration.
2. For long duration, it is appreciable.
3. The mass does not change.
2.2 Increasing gravitational force

The gravitational force at Earth’s surface is derived from Newton’s law:
\[ F = \frac{K(m1 \cdot m2)}{r^2}, \]
where \( K \) is a constant, \( m1 \) and \( m2 \) are the masses of the Earth and a rigid body respectively, and \( r \) is the distance of the two. By assumptions, \( m1 \) and \( m2 \) are constants but \( r \) may change very slowly in geological time. We will assume the rigid body has unit mass and \( p \) is a parameter equal to \( 1/r \). The equation is reduced to
\[ F = K(m \cdot p^2) \]  
(1)
Since \( r \) is shrinking, \( p \) is a monotonically increasing function of the geological time. In other words, at each experiment, we assume \( p \) is a constant on geological time. However for different instances (in the scale of geological time), we have different values of \( p \). Note that during the experiments, we use time too, we will refer it as local time. Local time is negligible in the scale of geological time.

Now let us back to pre-Galileo era, and assuming we are trying to discover the equation of motion by conducting some experiments: We throw a stone to the sky and kept records of the moving positions \( (x, y) \), where \( y \) is the height and \( x \) is the time. During this experiment \( p \) is a constant and the record is enter into a table. For different instance \( p \), we denote the table by \( EM(p) \).

3. NN IN DIFFERENT INSTANCES

In this section, we will train the neural networks using the tables listed in the Appendix: \( EM(p=0.6) \), \( EM(p=1.2) \), \( EM(p=5.0) \), \( EM(p=10) \), \( EM(p=20) \), and \( EM(p=267) \). In other words, the radii \( r \) are 1.6, 0.83, 0.2, 0.1, 0.05, and 0.003745 times of normal size.

The neural networks found have neurons 3, 4, 4, 5, 6, and 10. For almost all \( p \) the architectures are different; only at \( p=1.2 \) to 5.0 the architecture is the same. In other words, the NN is adaptable only during this periods.

3.1 The NN of some experiments

This section is import from [5]. We use Matlab (Version 5.2, Network Tool box version 2.0b) to produce a network for each data set \( EM(p) \).

3.1.1 Initialization of parameters

\( df = 10; \) % frequency of progress displays (in neurons).
\( me = 100; \) % maximum number of neurons.
\( eg = 0.02; \) % sum-squared error goal.
\( sc = 1; \) % spread constant radial basis functions.

% The training is done by
\[ [w1,b1,w2,b2,nr,err] = solverb(x,y,[df me eg sc]); \]
% The network is:
\[ ny = simurb(x,w1,b1,w2,b2); \]
For different \( p \), but fix \( df, me, eg, sc \), we train one neural network. For convenience, we write Radbas \((t) = e^{–t^2}\).
3.1.2 The case \( p=0.6 \)

The NN found has
1. Number of neurons = 3
2. Mean square error = 0.00000940
3. The activation functions are:
   
   \[
   \begin{align*}
   N_1 &= \text{Radbas}(0.832555|x-2.000000|) \\
   N_2 &= \text{Radbas}(0.832555|x-0.000000|) \\
   N_3 &= \text{Radbas}(0.832555|x-1.959184|)
   \end{align*}
   \]

4. The “output equation” of this neural network is:
   
   \[
   n_y = 1.581662 + 15.470798 N_1 - 1.089470 N_2 - 15.651405 N_3
   \]

3.1.3 The case \( p=1.2 \)

1. number of neurons = 4
2. mean square error = 0.00001787
3. The activation functions are:
   
   \[
   \begin{align*}
   N_1 &= \text{Radbas}(0.832555|x-2.000000|) \\
   N_2 &= \text{Radbas}(0.832555|x-0.000000|) \\
   N_3 &= \text{Radbas}(0.832555|x-1.387755|) \\
   N_4 &= \text{Radbas}(0.832555|x-0.040816|)
   \end{align*}
   \]

4. The “output equation” of this neural network is:
   
   \[
   n_y = 6.342275 - 0.863436 N_1 + 4.680662 N_2 - 3.558398 N_3 - 8.596080 N_4
   \]

3.1.4 The case \( p=5.0 \)

1. number of neurons = 4
2. mean square error = 0.00036011
3. The activation functions are:
   
   \[
   \begin{align*}
   N_1 &= \text{Radbas}(0.832555|x-2.000000|) \\
   N_2 &= \text{Radbas}(0.832555|x-0.000000|) \\
   N_3 &= \text{Radbas}(0.832555|x-1.020408|) \\
   N_4 &= \text{Radbas}(0.832555|x-0.040816|)
   \end{align*}
   \]

4. The “output equation” of this neural network is:
   
   \[
   n_y = 112.167935 - 53.678488 N_1 - 46.336989 N_2 - 56.277598 N_3 - 10.177362 N_4
   \]

3.1.5 The case \( p=10 \)

1. number of neurons = 5
2. mean square error = 0.00015827
3. The activation functions are:
   
   \[
   \begin{align*}
   N_1 &= \text{Radbas}(0.832555|x-2.000000|) \\
   N_2 &= \text{Radbas}(0.832555|x-0.000000|) \\
   N_3 &= \text{Radbas}(0.832555|x-1.020408|)
   \end{align*}
   \]

4. The “output equation” of this neural network is:
   
   \[
   n_y = 112.167935 - 53.678488 N_1 - 46.336989 N_2 - 56.277598 N_3 - 10.177362 N_4
   \]
N4 = Radbas(0.832555*|x-0.040816|)  
N5 = Radbas(0.832555*|x-1.959184|)  
4. The “output equation“ of this neural network is:  
\[ ny = 555.573432-1950.965374*N1-1799.408825*N2 
-96.463243*N3+1540.885483*N4+1706.926296*N5 \]

3.1.6 The case <p=20>

1. number of neurons = 6  
2. mean square error = 0.00031677  
3. The activation functions are:  
N1 = Radbas(0.832555*|x-2.000000|)  
N2 = Radbas(0.832555*|x-0.000000|)  
N3 = Radbas(0.832555*|x-1.020408|)  
N4 = Radbas(0.832555*|x-0.040816|)  
N5 = Radbas(0.832555*|x-1.959184|)  
N6 = Radbas(0.832555*|x-0.081633|)  
4. The “output equation“ of this neural network is:  
\[ ny = 2230.452479-7785.609358*N1-11858.062325*N2 
-1562.878035*N3 +15680.822505*N4 
+6793.726204*N5-4886.586480*N6 \]

3.1.7. The case <p=267>

For c = 267, the neurons number jump to 10. We skip the details.

3.2 Discussions

Readers may want to go to appendix to find the precise meanings of NN terminology.

1. If the “physics,” expressed in Eq(1), does not change much, the architecture of NN remains the same; see the change from p=1.2 to p=5.

In this case, the changes of weights (in NN) are adequate for adapting the changes of physics.

2. If the “physics” does change very much, for examples, p changes from 0.6 to 267, the number of neurons increases from 3 to 10. For such drastic changes, NN cannot adapt; see next section.

4. NEURAL NETWORK

4.1 Universal Approximations

An NN has been regarded as a black box that processes the input data and outputs a set of new values. Mathematically, an input and output defines a function. In other words, NN defines a parameterized function by “NN-geometry”; see Figure 1. The strength in this method is that geometry provides the mechanism for supporting infinitely many parameters, the weights and bias. By adjusting these weights and bias properly, NN can generate various useful functions.
4.1.1 Universal Approximation Theorem (UAT)

If we let the weights and bias run freely, the formula (A1) in the Appendix 2 generates a Schauder base for reasonable function space F, such as Lp-space. (A Schauder base, in this context, is a subset \( B \subseteq F \) such that any function in F can be approximated by finite linear combination of functions in B. B is a basis of Banach space). The formula (A2) gives a mechanism that one can approximate a given function (defined by training data) by a finite linear combination of Schauder basis. The approximation is done by adjusting the weights and bias.

4.1.2 Practical Realization of UAT

4.1.2.1 Initialization:

The users choose a set of parameters that is large enough to cover the application’s needs. Users’ initialization is a very important step; it essentially sets infinitely many parameters to zero. In the last example, that means to choose: \( df = 10 \); \( me = 100 \); \( eg = 0.02 \); \( sc=1 \). More explicitly, for example, the only possible non-zero neurons are 100. Any other parameters have already been set to zero.

4.1.2.2 Fine tuning:

The so called learning or training really means users determine (by back propagation and etc.) the precise values of the parameters using a given set of data, called training data.

4.2. NN-Learning and Adapting

Let us examine our example.

4.2.1 Training Data and “Spotted” Curve:

By recording the experiments, we have a numerical table, EM(p), that defines numerically the height \( y \) as a “function” of \( x \) (defined only on points in the table), where \( x \) denote the local time. The table, EM(p), plots a “spotted” curve on (x, y)-plane. This spotted curve only has finitely many points.

4.2.2 Patterns and “Continuous” Curve:

Mathematically NN is a parameterized function \( f(w,b)(x) \) defined analytically by activation functions with weights and bias \((w, b)\) as parameters [8]. To work on (x,y)-plane, we will use the graph \((x, f(w,b)(x))\) instead of function. \((x, f(w,b)(x))\) is defined analytically (in closed form), and will be referred to as analytic curve; it is a pattern derived from the training data. (There is no formal definition of patterns. Loosely a closed form formula that can fill in all the gaps of data is a pattern).

If we let the parameters \((w, b)\) runs freely, the universal approximation theorem (UAT) guarantees that the curve \((x, f(w,b)(x))\) moves by all curves in the state space (within the given -neighborhoods). Note that though the state space is finite dimensional, the space of all curves are infinite dimensional. UAT guarantees that the spotted curve will be covered by the trajectory.

It is important to note that at the beginning, there are infinitely many parameters in NN. However, in practical applications, we do not handle infinitely many parameters. By initialization, human experts often select a sufficiently large finite set. However, there is no uniform sufficiently large finite set for all applications.
4.3 Adaptive Dynamic Systems
Let us set up the environment:

4.3.1 Training Data EM(p):
Let p be the parameter, and assume for each p, there is a set of training data, EM(p). Since p is an increasing function of geological time; for convenient, we simply, by abuse of language that p is the geological time.

4.3.2 The pattern NN(p):
We will use NN(p) to denote the pattern derived from EM(p). NN(p) can be interpreted as NN with fixed values in the parameters, the function defined in closed form, or the analytic curve; they are all equivalent.

4.3.3 Key Questions:
Note that if p and p’ are near, the physics will tell us that EM(p) and EM(p’) are almost the same. Intuitively, NN(p) and NN(p’) should be nearly the same too, However, we do not have formal notion of “almost the same” in neural networks yet.

However, an interesting question is:
Could NN(p) transforms to NN(p’') by adjusting the parameters, when p moves to p’?
“Yes” has been believed by many authors. However, the examples in Section 3 clearly said that it couldn’t. However, we would like to know the mathematical reasons behind the examples. In next few paragraph, we will try to understand the reasons.

4.3.4 The dynamic system of dynamic systems
In the example (throwing a stone) the changing of “physics” (shrinking earth) moves the spotted curve in the state space. The changing of the “physics” itself is a dynamic system that pushes the “spotted” curves. We will call the trajectory physics-trajectory.
Next let us examine from NN side: The changes of weights and bias (the parameters of NN) moves the analytic curve. It is a dynamic system, We will call the trajectory NN-trajectory. It is clear the changes of weights and bias bear no physics.
Though two trajectories, NN-trajectory and physics-trajectory do intersect (guaranteed by UAT), but their flows of movements are not. This explains why NN(p) will not move to NN(p’’) by merely weights and bias’ changes.

4.3.5 Granular Neural Networks.
In examine the example, we propose to consider granular activation functions in the sense that each activation function belong to a granule. Each granule should consist of deformed functions of original activation functions; and deformed by the physical meaning (application semantics). For this example, it means we should define a neighborhood for each activation function, Radbas(t)=e–t^2, by deforming it according to the changing of “physics.” In this case, it seems impossible, since Radbas(t) bear no physical meanings. So in order for the NN to be able to adapt, the activation function have to carry some application semantics. Now if we interpret activation functions as W-sofsets, then the neighborhood is a qualitative fuzzy set that will be explained in next.
An NN with qualitative fuzzy sets as it activation function is called a Granular NN.

5. QUALITATIVE-FUZZY LOGIC

To avoid confusing, we will adopt the following convention: Following [10], the traditional fuzzy set will be referred to by a technical term **W-sofset**, which is defined solely by a membership function [13, pp.12]. The term "fuzzy set" will have no technical content; it refers to the intuitive vague concept. The term qualitative fuzzy set refers roughly to a mathematical object that is defined by a set of membership functions. Its precise meaning is depended on how such a set is defined [12, 11,10, 6, 4,3].

5.1 Traditional FL

Traditional fuzzy logic has 4 steps:
1. A set of linguistic rules.
2. Each linguistic variable is assigned a W-sofset (fuzzification). We will regard the linguistic variable as the name of the w-sofset. This assignment turns the linguistic rules into fuzzy rules.
3. By defuzzification (e.g., Madani inference), we produce a potentially the desirable function
4. By tuning, the potentially desirable function may be adjusted to the final target function.

We would like to offer some mathematical comments:
1. Madani's inference method expresses the target functions as linear combination of W-sofsets (its membership function).
2. Kosko collects all those membership functions (of w-sofsets) into a library. His library is a Schauder base of the membership function space [2]

5.2. FL on Qualitative Fuzzy Sets

The only change we will do is at step 2 assigning, instead of w-sofsets, qualitative fuzzy sets to linguistic variables. Moreover, we stress that the expert should pay great attention to see if the qualitative fuzzy set do carry some semantic of physical world; this was one of the motivation that we develop qualitative fuzzy sets along the "pulling or stretching" [3,4,6,9]

Note that each qualitative fuzzy set is defined by a set of membership functions, we pick a representative to participate in step 3 and 4. At tuning time, we adjust the membership function within the qualitative fuzzy set.

6. CONCLUSIONS

In non-linear controls, NN and FL have been used to capture the control functions. If the control system is reasonably stationary in the sense that the ranges of fluctuations are small, then NN/FL may be able to manage adequately. However, if the dynamic has a large range of fluctuations, then traditional NN/FL may not be adequate. We need deeper universal approximators. Earlier, we propose the notion of semantic oriented approximators [7, 5]. That is, one should search for an approximation theory that can approximate a family of "continuously deforming" functions by an "adaptable" NN/FL. In this paper, we propose to replace traditional fuzzy sets (technically W-sofsets) by qualitative fuzzy sets (a granule of membership functions) and activation functions by activation granules.
APPENDIX 1

The Table EM(p) is generated theoretically by Matlab using
x=0:2/49:2; % The expression means the interval [0, 2] is divided into 49 intervals (50 equal distance
points).
y=a*(x -1).^2+v*(x -1)+h, % This is a function with following meaning: a is the acceleration, v the
initial velocity, and h the initial height
The initial setting: h=0.5, v=0.5, and a = p2, where p varies.

REFERENCES

Programming Techniques.” Addison-Wesley Publishing Co.
2359-2363
Neural Networks, Bulletin of International Rough Set Society, Vol I, No1, March, 1999, Vol 3, No 1-2,
7 – 10
Journal of Approximate Reasoning, Vol 15, No 4, 1996,3 95-414. also "Rough-Fuzzy Controllers for
Networks.” Neural Computation 3, 246-257.
on Multiple-Valued Logic, Tokyo May 20-22, 1999
Wailea, Maui, Hawaii, June 11-16,2000
1991. r