A&QT-R 2002 (THETA 13)
2002 IEEE-TTTC International Conference on Automation, Quality and Testing, Robotics
May 23 – 25, 2002, Cluj-Napoca, Romania

The Nelder - Mead Method in Optimization Using Artificial Neural Network

Kakucs András, Ph.D.

INDELCO srl, Tg.Mureş, Romania, e-mail: kakucs_a@ibcnet.ro

The task of planning and designing is often complex because there is often no model or detailed understanding of how changes in the parameters affects product properties. Traditional approaches include statistical methods and experimentation. Both methods are expensive. In addition, in most cases the conventional, linear statistical tools do not work at all.

A significant opportunity exists to improve operations and resulting profitability by streamlining the design task. Artificial neural network approximation addresses this opportunity which is most useful in an environment where theoretical descriptions are difficult to obtain, but partial knowledge about the process is known and input-output data are available. The obtained relation (the trained artificial neural network) is then used as an interpolating function to estimate product performance when given specific parameters (direct modeling). The trained artificial neural network is used as the object function of a Nelder-Mead simplex to optimize parameters to accomplish desired product characteristics (inverse modeling or optimization).

Keywords: artificial neural network, optimisation,

1. INTRODUCTION

The human brain is constituted by special, interconnected cells called *neurons*. Biological neurons transmit electrochemical signals over neural pathways. Each neuron receives signals from other neurons through special junctions called *synapses*. Some input tend to excite the neuron, others tend to inhibit it. When the cumulative effect of the inputs exceeds a threshold, the activated neuron sends a signal toward the other neurons.

The artificial neuron models this simple behavior: each artificial neuron receives a set of inputs. These are weighted and their sum determines the activation level. The artificial neural network contains a large number of simple neuronlike processing elements and a large number of weighted connections between elements. Each neuron is excited by a set of weighted inputs. This excitation determines the activation level of that neuron (its output). Their response constitutes the excitation of the neurons from the next layer. The activation level of the neurons from the last layer (of the output units) determines the response of the neural network to the given excitation.

The process trough the values of the weighting coefficients are obtained is the *learning* or *training* phase: starting from some arbitrary or random weight setting the neural network is trained to adapt itself to the characteristics of the training instances. In each

training cycle the obtained response is compared with the desired one. The weighting coefficients are modified to minimize the difference between the obtained and desired output.

In our test programs we used total connected feedforward neural networks with one hidden layer. Biasing was introduced to reduce the effect of the overfitting.

2. THE NELDER-MEAD DOWNHILL SIMPLEX OPTIMIZATION METHOD

The multidimensional optimization is the problem of finding the minimum of a function of more than one independent variable. The Nelder - Mead method was adopted in our studies because it requires only function evaluation (that is performed using the trained artificial neural network), not derivatives (those are unknown).

A *simplex* is the geometrical figure, in n dimensions consisting of $n + 1$ vertices (points) and all their interconnecting entities (segments, faces etc.). A 3D simplex is a tetrahedron (Fig. 1).

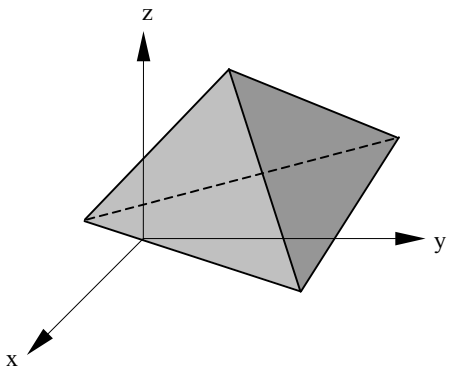

Fig. 1. The 3D simplex (the initial simplex)

In one-dimensional minimization it is possible to bracket a minimum, so that the success of a subsequent isolation is guaranteed. In a multidimensional space there is no analogous procedure. So given the algorithm starts with a guess that is an n dimensional simplex.

This guess is the initial simplex defined in the middle of the field of the input variables. As it was shown, the initial simplex is build up from $n + 1$ points. If $\boldsymbol{P}_0$ is considered as the initial starting point, the other n points can be taken as

$$\boldsymbol{P}_i = \boldsymbol{P}_0 + \lambda \cdot \mathbf{e}_i, \qquad (1)$$

where $\mathbf{e}_i$ are unit vectors (of the nD space) and λ is a constant.

In our case the number of the dimensions n is the number of input variables and the vertices of the simplex will be $n + 1$ input data sets of the artificial neural network.

The algorithm is then supposed to make its own way downhill through the n-dimensional topography, until it encounters a minimum (that can be a local minimum) of the studied function.

This function in our case is the relation modeled by the trained artificial neural network. If we have a single output property the value of the function is simply the response of the trained network (the estimated output property). When we have more output

properties, the value of the function to be minimized is obtained as the sum of the estimated outputs.

The downhill simplex method now takes a series of steps, most of them just moving the point of the simplex where the function is largest (this is the *highest point*) trough the opposite face of the simplex to a lower point. These steps are called *reflections* and they are constructed to conserve the *volume* of the simplex.

When it can do so, the method expands the simplex in one or another direction to take larger steps. When it reaches a *valley floor*, the method contracts itself in the transverse direction and tries to ooze down the valley.

In the situation when the simplex is trying to pass trough a narrow place, it contracts itself in all directions around of its lowest point.

The terminating criteria is the comparison of the *distance* of the moving is smaller than some tolerance, or alternatively, of the decrease of the function value in the terminating step. For a certain result is a good idea to restart the minimization routine at a point where a minimum was found.

3. CONSTRAINED OPTIMIZATION

The role of the optimization is to generate a set of *input properties* in response to the specified set of *output properties*.

Their range can be constrained to any subinterval but they ought to be covered by the original data (used in training). Constraining the definition range of the input properties is rather simple; e.g. if the chosen evaluation point has a coordinate over of the valid range its value is set to the upper limit of this interval.

Constraining the range of the output properties can be resolved using penalty factors: if one estimated output property is out of the constrained range it is set to a high value (the algorithm searches the smallest possible value over this range).

The rank (importance) of the output properties can be set according *priority* values: higher values mean higher priorities. These priority values act as weighting coefficients and the values of function to be optimized will be the weighted sum of the estimated output properties.

The algorithm presented till now searches the *minimum* of the studied function (relation), but the *optimum* does not mean always this minimum. So given, there were introduced three types of optimization:

- 1. tent optimization - the value of this output property is expected as close to the mean value of the limits of the definition interval (is its middle) as possible. The quantity to be minimized is the absolute value of the difference of the estimated output and of the mean value of the constrained interval.
- 2. uphill optimization - the value of this output property is expected as close to the upper value of property domain as possible. The quantity to be minimized is the absolute value of the difference of the estimated output and of the upper limit of the constrained interval.
- 3. downhill optimization - the value of this output property is expected to be as close to the lower value of property domain as possible. The quantity to be minimized is the

absolute value of the difference of the estimated output and of the lower limit of the constrained interval.

Other available constraints are the prescription of the ratio of the input properties: in other words, between the constrained coordinates of the vertices of the simplex must exist a constant proportionality.

The sum of the input properties also can be constrained. If the taken vertex has the sum of the coordinates over of the allowed range its coordinates are reduced using the same scale factor to get their sum as the upper limit of the definition interval.

4. VALIDATION

To have the possibility to control how the used algorithm works we considered a large number of 'measured' data points (sets) obtained using a known functions, as:

$$z = \cos(x) + \sin(y) + y / 2 \qquad (2)$$
$$x = 2\,\pi\,a;\ y = \pi\,a;\ a \in [0, 1].$$

It is easily observed that this is a transcendent function that is the equation of a surface represented in the next picture (Fig. 2). It is known that modeling of such functions using polynomials is rather difficult.

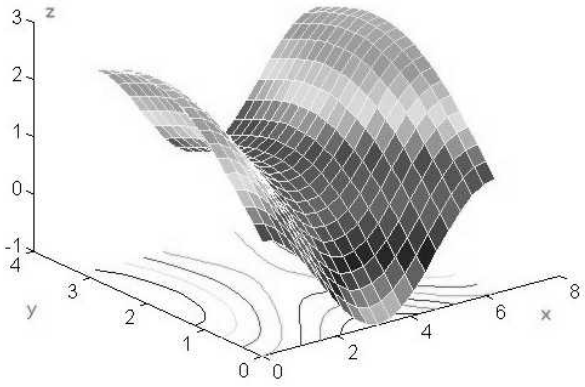

Fig. 2. The exact shape of the studied function

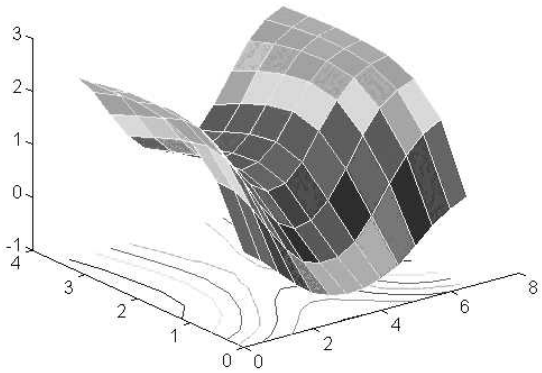

Fig. 3. The image of the studied function seen by the trained neural network

The 'measured' (x, y, z) sets were obtained as values of the function at uniformly distributed random (x, y) points. The previous picture (Fig. 3) is the image of the $z = f(x, y)$ surface as it is 'seen' by the neural network. This plot was obtained using estimated z values for $x \in [0, 6.5]$ and $y \in [0, 3.5]$, varying them by 0.5.

Comparing these two figures we can observe a good matching, excepting the boundaries of the represented domain (they were not covered by the 300 sample points).

The obtained neural network was used in inverse modeling. The aim of the optimization was to find the values of the input properties those gives the desired output properties, respecting some conditions (Fig. 4):

The carried-out results were obtained as: $x = 1.333$, $y = 0.667$, $z = 1.164$.

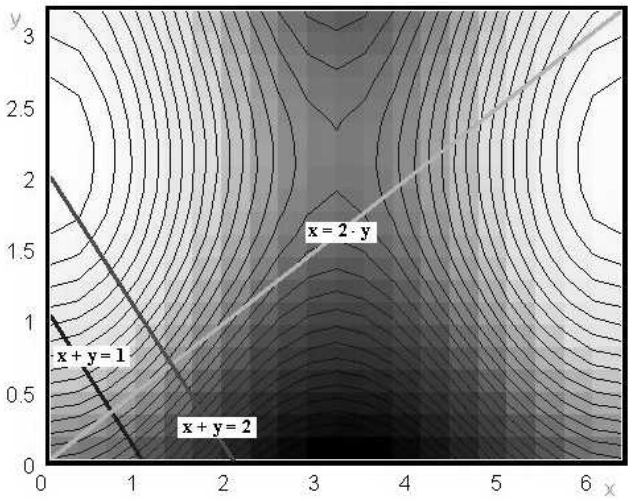

Fig. 4. Interpretation of the constraints:
- the sum of x and y must be $(x + y) \in [1,2]$
(this domain is limited by the two parallels)
- the ratio between x and y must be $x / y = 2$
(this constrain is shown by the light diagonal line)
- downhill optimization for z
(the lowest possible value of z is searched)
Darker colours mean lower values of z.

Let verify if the initial conditions were satisfied:
- making some estimations near of these values (of course: respecting the initial conditions) we will obtain values of z higher than 1.164;
- x : y = 1.333 : 0.667 = 1 : 2; - the prescribed ratio is obtained
- x + y = 1.333 + 0.667 = 2.0 - inside of the constrained interval of sums.

5. AN APPLICATION

We have studied the mechanical properties of the jumping joints obtained in the same conditions using eight different welding rod steels and the obtained mean values are summarized in the next table:

Components	C	%	0.08	0.09	0.08	0.08	0.08	....
	Mn	%	0.55	0.6	0.6	0.65	0.5	
	Si	%	0.2	0.25	0.4	0.35	0.3	
	S & P	(over 0.04 %)	no	no	yes	yes	yes	
Tensile strength	l_1	N/mm^2	510	535	525	525	535	
Ductility limit	l_2	N/mm^2	430	450	465	465	465	
Breaking strain	d	%	26	27	28	28	26	

Each steel is an alloy of the following elements: Fe (iron), C (carbon), Mn (manganese) and Si (silicon), some of them having S (sulfur) and P (phosphorus) impurities in significant quantities. The "no" and "yes" logical entries were replaced by "0" and "1" numerical ones.

The relation between the composition and the mechanical properties of these steels was given by a neural network with the relative errors of 0.81% (tensile strength), 1.51% (ductility limit) and 2.99% (breaking strain).

Let suppose that we have to determine the composition of steel unpurified with P and S that assures ductile jumping joints with the possible highest mechanical properties. The ratio of the components must be Mn : Si = 2.5 : 1. The sum of the alloying elements must be under 1.2%.

To resolve this problem we have to make an optimization (inverse modeling) in the next conditions:
- constrain the "S & P" property on the [1, 1] interval (it is forced to 1);
- set the ratio value 2.5 for Mn and 1.0 for Si;
- set the sum of the C, Mn and Si components to [0.78, 1.2];
- set the kind of optimization to 'uphill' for *l1* and *l2* (possible highest mechanical properties);
- set the kind of optimization to 'uphill' for *d* (ductile steel).

The obtained properties will be:
- steel with components C = 0.08%, Mn = 0.65 %, Si = 0.26 %, unpurified with S and P, with *d* (Breaking strain) = 26.9 %, *l1* (Tensile strength) =533 N/mm^2, *l2* (Ductility limit) = 458 N/mm^2.

6. CONCLUSION

Often there is no model or detailed understanding of how changes in formulation and processing conditions affect product properties. Many times the designing engineer is forced to rely on experience, insight and trial-and-error to find a formula that is sufficient, if not optimal.

The presented algorithm minimizes the need for analysis and experimentation typically required for new product design. It builds upon conventional techniques by integrating neural networks and optimization methods. It is most useful in an environment where theoretical descriptions are difficult to obtain, but partial knowledge about the process is known and input-output data are available.

References:

1. M. W. Firebaugh (1989)- **Artificial Inteligence - A Knowlidge-Based Approach** - PWS-Kent Publishing Co. Boston
2. S.L.S. Jacoby, J.S. Kowalik, J.T. Pizzo (1972)- **Iterative Methods for Nonlinear Optimization Problems** - Prentice-Hall
3. Bart Kosko (1992)- **Neural Networks and Fuzzy Systems** - Prentice-Hall
4. Li Min Fu (1994)- **Neural Networks in Computer Science** - McGraw Hill
5. M.J. Norusis (1985)- **Advanced Statistics Guide** - McGraw Hill
6. W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery (1994)- **Numerical Recipes in Fortran** - Cambridge University Press