Artificial Stupidity: A Reply

Murphy, Koehler, and Fogler [1997] gave in the last issue of the *Journal of Portfolio Management* an account of how to raise a neural net's IQ. The purpose of this reply is to point out some of the general difficulties with neural nets. Also, I would like to mention an alternative method, namely Pade approximants, which does not suffer from these difficulties.

Murphy, Koehler, and Fogler [1997] document that even approximating a rather simple nonlinear function in one dimension through neural nets requires a considerable amount of fine-tuning. In particular, they are faced with a time-consuming and complex algorithm where a number of rather delicate parameters have to be iteratively adjusted in order to yield an acceptable neural net. A considerable caveat is also that the available data has to be split further into a training set (40 values in their example) and a holdout set (160 values). Different neural nets are constructed using the training set and then tested against the holdout set. This methodology is inherently flawed since it amounts to an in-sample test. A true out-of-sample procedure would require

to use only the training set, decide on the best model, and then check that model once against the holdout set. Given the rather poor fit of their neural nets on the boundaries, even for their best model, the true out of sample performance is questionable at best.

But at times going back to the basics can yield superior results. Murphy, Koehler, and Fogler [1997] rightly notice that a polynomial fit does poorly. However, even a standard reference work such as "Numerical Recipes", Press et al. [1992, pp. 104-107, 194-201], suggests Pade approximants (also known as rational functions approximation) as the method of choice for nonlinear relations. The idea is to approximate the unknown function by the ratio of two polynomials:

$$f(x) = \frac{a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m}{1 + b_1 x + b_2 x^2 + \ldots + b_n x^n} \qquad (1)$$

We can rewrite equation 1 slightly and immediately see that the following system of equations can be conveniently solved by Ordinary Least Squares algorithms:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m - b_1 x f(x) - b_2 x^2 f(x) - \ldots - b_n x^n f(x)$$

(2)

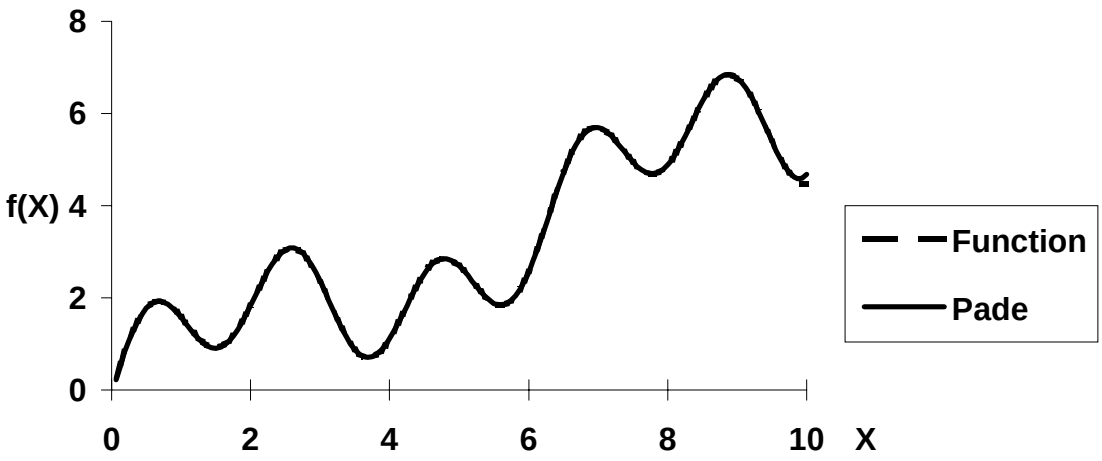
The only choice parameters here are the number of terms, m and n. The easiest is to start with m=n=0 and increase those values if the fit is insufficient. A useful tool is to check for poles where f(x) goes to $\pm\infty$. We often require that there are no poles within the region of interest and can insure that by restricting m and n accordingly. Also, n should be chosen as m+1 if we know that f(x) approaches 0 for large values of x. Finally, boundary conditions can be implemented by simply adding sample values of x and f(x) on the boundary to the training set.

Using a similar training set and the same function as Murphy, Koehler, and Fogler [1997], I obtain an almost perfect fit throughout the holdout set by using m=n=10, as is shown in figure 1. Only a small discrepancy is noticeable at the right boundary. In particular, this approach uses only the training set repeatedly as m and n vary and then performs only one final check against the holdout set.

EXHIBIT 1

FUNCTION: f(x) = sin(x) + sin(3x) + 0.6x

VS. PADE APPROXIMANT



This reply might serve as a cautionary tale that mainstream methods for nonlinear problems can be very useful. Jumping onto the bandwagon of implementing neural nets would then require a careful assessment that alternative methods are exhausted. However, from my own research of notoriously nonlinear problems in the fields of options and probability, I still have not reached that point.

References

Murphy, Christopher M., Gary J. Koehler, and H. Russell Fogler. “Artificial Stupidity.” *Journal of Portfolio Management*, Winter 1997, pp. 24-29.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in FORTRAN: the art of scientific computing*, 2nd. Ed. Cambridge: Cambridge University Press, 1992.