CERN LIBRARIES, GENEVA

IC/94/229

SCAN-9411056

SCW 94 45

INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS

A MODIFIED BACKPROPAGATION ALGORITHM FOR TRAINING NEURAL NETWORKS ON DATA WITH ERROR BARS

INTERNATIONAL
ATOMIC ENERGY
AGENCY

UNITED NATIONS
EDUCATIONAL,
SCIENTIFIC
AND CULTURAL
ORGANIZATION

Klaus A. Gernoth

and

John W. Clark

MIRAMARE-TRIESTE

International Atomic Energy Agency
and
United Nations Educational Scientific and Cultural Organization
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS

**A MODIFIED BACKPROPAGATION ALGORITHM
FOR TRAINING NEURAL NETWORKS
ON DATA WITH ERROR BARS**

Klaus A. Gernoth
International Centre for Theoretical Physics, Trieste, Italy

and

John W. Clark
McDonnell Center for the Space Sciences and Department of Physics,
Washington University, St. Louis, MO 63130, USA.

MIRAMARE - TRIESTE

August 1994

ABSTRACT

A method is proposed for training multilayer feedforward neural networks on data contaminated with noise. Specifically, we consider the case that the artificial neural system is required to learn a physical mapping when the available values of the target variable are subject to experimental uncertainties, but are characterised by error bars. The proposed method, based on a maximum likelihood criterion for parameter estimation, involves simple modifications of the on-line backpropagation learning algorithm. These include incorporation of the error-bar assignments in a pattern-specific learning rate, together with epochal updating of a new measure of model accuracy that replaces the usual mean-square error. The extended backpropagation algorithm is successfully tested on two problems relevant to the modeling of atomic-mass systematics by neural networks. Provided the underlying mapping is reasonably smooth, neural nets trained with the new procedure are able to learn the true function to a good approximation even in the presence of high levels of Gaussian noise.

1. Introduction

Artificial neural networks with varied dynamics, architectures, and learning rules are emerging as versatile and powerful tools for the solution of a broad range of problems involving optimization, classification, function approximation, and model building [1,2]. In addition to countless technological applications with potential commercial impact, recent years have witnessed a proliferation of intriguing and fruitful scientific uses of the collective computational properties of these systems. To give just a few examples of the latter activity, neural nets have been developed for:

- (i) Prediction of secondary and tertiary (folded) structure of proteins from their primary amino-acid sequence [3],
- (ii) Elimination of atmospheric phase distortion from astronomical observations with adaptive optics [4],
- (iii) Track reconstruction, event classification, and triggering in high-energy physics [5],
- (iv) Learning and prediction of the systematics of atomic nuclei, including masses, neutron separation energies, spins, and branching ratios for decay channels [6-9].

Most of the current scientific applications (e.g. (ii) and (iii)) are seen in experimental settings and involve neural networks in data management, reduction, or analysis, sometimes in real time. In these cases a network has the task of interpreting individual experimental observations or identifying interesting objects, events, features, or properties. However, some of the applications (notably (i) and (iv)) are more theoretical in character. Specifically, the artificial neural assembly may be required, under training with a subset of the existing data base for a given class of physical systems, to create a phenomenological model of these systems, for subsequent use in prediction. Conceptually, such a “neural network phenomenology” recapitulates essential steps of the human theoretical program, namely the extraction of regularities or correlations from experimental observations, the creation of a model that expresses these regularities, and the comparison of consequent theoretical predictions with results from new experiments. As with conventional phe-

nomenological treatments, an essential requirement for a successful neural network model is **generalization ability**, reflected in accurate prediction for cases absent from the training set.

More abstractly, artificial neural networks, as they are employed in both technology and science, may be regarded as devices for performing statistical inference. They provide an attractive nonparametric approach to estimation of *a posteriori* probabilities. In favorable circumstances, these systems can approach the Bayesian ideal (see e.g. refs. [9-12]).

In the present paper we shall focus on the use of neural networks to perform a non-linear regression analysis on a given set of data consisting of pairs of sample inputs $\mathbf{x}$ and target responses $\mathbf{Y}$. In practice, the regression analysis is carried out by supervised training of a multilayer feedforward network [1], which involves the choice of a cost function that measures the overall error of network response and the minimization of this objective function with respect to the connection-weight parameters characterizing the network model. The most popular training algorithm is backpropagation [13], which is based on an incremental gradient-descent technique.

In dealing with experimental data, or data generated by stochastic computer simulation, one must confront the question of how experimental or statistical errors will affect the learning process and therefore the model that results from the training procedure. It is certainly undesirable for the network to learn (and hence fit) spurious details produced by such errors. On the other hand, one would like to make some use of the information that is contained in noisy data points. Here we shall propose and test a simple modification of the standard backpropagation algorithm that allows highly effective modeling to be achieved even when the output variables - and thus the nominal targets for learning - are afflicted with noise. The new algorithm, which is based on the well-known maximum-likelihood criterion [14] and adapts an earlier treatment of Möller and Nix [15] to neural-network analysis, is predicated on a prior assignment of “error bars” to the data points r that have been furnished for analysis. To be definite, we assume that the experimental

(or stochastic-computational) errors are normally distributed with zero mean and known standard deviations σ_r , although the approach can be extended to other characterizations of the noise.

The new learning algorithm is formulated in general terms in section 2. In section 3, its behavior is explored within the specific context of the modeling of atomic masses. In the primary example, relatively small multilayer feedforward nets are asked to capture the semi-classical liquid-drop mass formula in the presence of noise that is elevated to high levels.

The proposed modification of backpropagation learning is expected to prove valuable not only within the contexts of neural network phenomenology and experimental data analysis, but also in diverse technological applications where target variables are subject to noise.

2. Modified backpropagation algorithm (“cleanprop”)

In abstract terms, the problem that concerns us is to reconstruct a mapping from one set of variables (the “inputs”) to another set (the “outputs”), based on a restricted set of instances of the mapping, these instances being corrupted by noise. Let the mapping of interest be defined by a real function $Y(\mathbf{x})$ of a set of real input variables, $\mathbf{x} = x_1, \dots, x_I$. For our purposes it is useful to regard Y as a physical property that is determined, either through some natural law or through a corresponding “true” theory, by specification of the values of $x_1, \dots, x_I$.

Ideally, a set of n noise-free instances $(\mathbf{x}^{(r)}, Y(\mathbf{x}^{(r)})) = (\mathbf{x}^{(r)}, Y^{(r)})$, $r = 1, \dots, n$ (“the data”) is provided for the modeling process. One then seeks to develop a model of the underlying physical or theoretical mapping by training a multilayer feedforward neural network to fit the data. For a given training example $(\mathbf{x}^{(r)}, Y^{(r)})$, a coded version of $\mathbf{x}^{(r)}$ (“pattern” r) is impressed on the activities of the units of the input layer, and the result of the network’s computation is read off from the activities of the units of the output layer and compared with a coded version of the target output $Y^{(r)}$. A learning algorithm is applied to make incremental adjustments of the weights of the connections in the network to reduce the fitting error. Once this process has achieved a satisfactory fit of the training data, it is expected (or hoped) that the network will produce responses closely approximating the $Y^{(r)}$ values corresponding to test inputs $\mathbf{x}^{(r)}$ that do not occur in the training set. The neural network model does not qualify as an acceptable model of the underlying physical law or theory unless it performs well on this latter task - i.e., unless it is able to generalize from the examples it has learned.

The problem addressed in this paper departs from this ideal scenario in that only noisy data are available for the modeling process. The data consist of pairs $(\mathbf{x}^{(r)}, y_e^{(r)})$ with

$$y_e^{(r)} = Y^{(r)} + \epsilon_e^{(r)} \quad , \quad (1)$$

where $\epsilon_e^{(r)}$ is a random variable corresponding to experimental or theoretical error. (Here, “theoretical errors” will refer to deviations from the true theory that are present in the data points provided for the modeling process. For example, data from a Monte Carlo simulation carry statistical uncertainties.) If a network is trained using the standard learning algorithms (most notably, the backpropagation procedure), there is a danger that the net will learn the stochastic aspects of the data at the expense of the underlying regularities. Thus, the influence of the noisier data points should be suppressed in making adjustments of connection strengths during the learning phase, or, in more general terms, the poorest data should be given the least weight in the fitting process. We shall do this in a systematic manner by invoking established methods of statistical analysis and adopting a modified objective function or cost function in the training of feedforward network models. Although the arguments are framed in the language of neural nets, it should be evident that they are applicable more broadly to interpolation strategies based on minimization of an objective function with respect to model parameters.

Three key assumptions are made concerning the statistics of the problem:

- (I) The model of the mapping $\mathbf{x} \rightarrow Y$ given by the neural network (or other adaptive system) differs from the true mapping – from the true theory or from nature – in a large number of small terms having fluctuating signs and magnitudes [15]. Specifically, for the network outputs corresponding to the training examples we have

$$y_m^{(r)} = Y^{(r)} + \epsilon_m^{(r)}, \quad r = 1, \dots, n, \quad (2)$$

where the discrepancy $\epsilon_m^{(r)} \in N_m(0, \sigma_m^2)$ is a normally distributed random variable with zero mean and standard deviation σ_m .

- (II) The experimental or theoretical data $y_e^{(r)}$ supplied for Y at given $\mathbf{x}$ values contain stochastic errors $\epsilon_e^{(r)} \in N_e(0, (\sigma_e^{(r)})^2)$ that are also randomly distributed with zero mean, but with instance-specific standard deviations $\sigma_e^{(r)}$ (cf. Eq. (1)).
- (III) The input variables of the problem, $\mathbf{x}$, are not subject to noise.

Our purposes are served by shifting from the putatively natural view and regarding the true value $Y^{(r)}$ in (2), rather than the model output $y_m^{(r)}$, as the random variable. This accords with the fact that the model, by construction, gives a definite output value for each input $\mathbf{x}^{(r)}$. (Möller and Nix [15] offer a particularly vivid description of the operational situation: God uses the phenomenological model fashioned by his human creations, but with intrinsic parameters He decides (and which we must estimate); to the resulting $y_m^{(r)}$ He adds a random number $-\epsilon_m^{(r)} \in N_m(0, \sigma_m^2)$ to achieve a result $Y^{(r)}$ to His liking.)

Consequent to these assumptions and this view, the conditioned probability density for obtaining the value ξ for the random variable $y_e^{(r)}$ corresponding to input $\mathbf{x}^{(r)}$ may be written as

$$f_{y_e^{(r)}}(\xi | y_m^{(r)}, (\sigma_e^{(r)})^2 + \sigma_m^2) = \frac{1}{\sqrt{2\pi}} \frac{1}{\sqrt{(\sigma_e^{(r)})^2 + \sigma_m^2}} \exp \left[-\frac{(\xi - y_m^{(r)})^2}{2((\sigma_e^{(r)})^2 + \sigma_m^2)} \right]. \quad (3)$$

In this expression, $y_m^{(r)}$ is to be treated as a parameter of the distribution, along with the quantity σ_m (which will ultimately be used to characterize the fit achieved by the model). In turn, $y_m^{(r)}$ is determined by the intrinsic parameters ω_p , $p = 1, \dots, P$, that specify the particular model being applied (the one selected and used by God, in the Möller-Nix parable). In the present context, these intrinsic parameters, for a feedforward network model with given architecture and given forms for the unit activation functions, are of course just the weights of the modifiable connections between units (including the biases or thresholds of the units in hidden and output layers).

The problem that faces us is now a familiar one, namely that of estimating the unknown parameters (specifically, the model error measure σ_m and the fitting parameters ω_p) of a known statistical distribution (specifically, Eq. (3)). Accordingly, we appeal to a widely used prescription for parameter estimation, *the maximum likelihood criterion* [14,16,17]. It is supposed that the existing data is in fact the data that has the maximum likelihood of being obtained, according to the given distribution. For n data points, i.e. n “experimental” results $y_e^{(r)}$ for different values $\mathbf{x}^r$ of the driving variables, the likelihood

function is formed as the product of densities of the type (3):

$$L(\xi^{(1)}, \dots, \xi^{(n)}) = f_{y_e^{(1)}}(\xi^{(1)} | \sigma_m^2, \{\omega_p\}) \times \dots \times f_{y_e^{(n)}}(\xi^{(n)} | \sigma_m^2, \{\omega_p\}). \quad (4)$$

(Here we display only the essential conditioning parameters. The quotes on “experimental” mean that the data could be either experimental *or* theoretical in character, depending on the application.) Maximizing $\log L$ with respect to the model parameters $\{\omega_p\}$ and σ_m , we have $P + 1$ equations to be solved for these $P + 1$ quantities:

$$\sum_{r=1}^n \frac{y_e^{(r)} - y_m^{(r)}}{(\sigma_e^{(r)})^2 + \sigma_m^2} \frac{\partial y_m^{(r)}}{\partial \omega_p} = 0 \quad (i = 1, \dots, P), \quad (5)$$

$$\sum_{r=1}^n \frac{(y_e^{(r)} - y_m^{(r)})^2 - [(\sigma_e^{(r)})^2 + \sigma_m^2]}{[(\sigma_e^{(r)})^2 + \sigma_m^2]^2} = 0. \quad (6)$$

In Bayesian terms, this optimization strategy effectively assumes uniform priors for the parameters being determined. The properties of the maximum likelihood criterion are discussed at length in basic texts on mathematical statistics, for example Cramér [14] and von Mises [16]. Technically, it provides a consistent estimate, which, however, may be biased. Bias may be reduced by taking a larger data sample, if that option is practicable. Bias introduces a fractional error in the estimated parameter of order P/n , where P is the number of adjustable (intrinsic) model parameters and n is the size of the data sample.

As is readily seen, solution of (5) and (6) for the model error measure σ_m and the intrinsic model parameters ω_p is equivalent to minimizing the objective function

$$E = \frac{1}{2} \sum_{r=1}^n \frac{(y_e^{(r)} - y_m^{(r)})^2}{1 + (\sigma_e^{(r)}/\sigma_m)^2} \quad (7)$$

with respect to the ω_p (while holding σ_m fixed at the value determined by Eq. (6)) and solving Eq. (6) for σ_m (the $y_m^{(r)}$ that enter being evaluated at the minimizing set of ω_p).

It is advantageous to cast Eq. (6) into a form conducive to iterative solution [15]:

$$\sigma_m^2 = \frac{1}{B} \sum_{r=1}^n b_r [(y_e^{(r)} - y_m^{(r)})^2 - (\sigma_e^{(r)})^2], \quad (8)$$

where

$$B = \sum_{r=1}^n b_r, \quad b_r = [(\sigma_e^{(r)})^2 + \sigma_m^2]^{-2}. \quad (9)$$

One may start the iteration with the simple root-mean-square (rms) measure of model error, commonly adopted in the noise-free case:

$$\sigma_m^{[0]} = \left[\frac{1}{n} \sum_{r=1}^n (y_e^{(r)} - y_m^{(r)})^2 \right]^{1/2}. \quad (10)$$

Having brought the analysis to this stage, it is easy to see how the standard back-propagation algorithm (“vanilla backprop”) can be generalized to deal with noisy data. To frame the new procedure, we streamline the above notation, adapting it more closely to the model framework of feedforward neural networks. In practice, we work with coded versions of the quantities $\mathbf{x}$, Y , y_e , and y_m rather than the variables themselves. Accordingly, in applying the foregoing analysis we identify $y_e^{(r)}$ with the target activity value $t^{(r)}$ for a single output unit that delivers, in coded form, the result of the network’s computation of the variable Y . Similarly, $y_m^{(r)}$ is identified with the actual output of this unit, $a^{(r)}$, upon presentation of pattern r , i.e., when the input-layer units are activated so as to encode $\mathbf{x}^{(r)}$. We further introduce the abbreviations $\sigma_e^{(r)} = \sigma_r$ and $\sigma_m = \sigma$ and define a cost function

$$E = \sum_{r=1}^n E_r \quad (11)$$

with pattern-specific error terms

$$E_r = \frac{1}{2} \frac{(t^{(r)} - a^{(r)})^2}{1 + (\sigma_r/\sigma)^2}. \quad (12)$$

The cost function $E^{[0]}$ of conventional backprop has the same form, but with unity in place of the denominator appearing in Eq. (12). As with vanilla backprop, the nominal goal of the modified backpropagation scheme is to minimize the pattern-summed cost function by incremental adjustment of the connection weights w_{ij} characterizing the network, thereby determining the intrinsic parameters ω_p . The new feature is that as the minimization based

on Eqs. (11)-(12) progresses, the unknown model-error parameter σ in the denominator of E_r is to be re-determined consistently by solution of

$$\sigma^2 = \sum_{r=1}^n c_r [(t^{(r)} - a^{(r)})^2 - \sigma_r^2] , \quad (13)$$

where c_r is defined by

$$c_r = (\sigma_r^2 + \sigma^2)^{-2} / \sum_{s=1}^n (\sigma_s^2 + \sigma^2)^{-2} . \quad (14)$$

Again as in ordinary backprop, the weight parameters explicitly enter the objective function E of Eq. (11) through the functional dependence of the activities $a^{(r)}$ present in Eq. (12). However, E also carries an implicit dependence on the weights through the parameter σ , since Eq. (13) also involves the $a^{(r)}$. Indeed, Eq. (13) gives explicit expression to the notion that the experimental errors should somehow be “subtracted out” in evaluating the model quality index σ .

At this point it is well to review some terminology and establish some conventions [1,2]. The activity (or output) a_l of a given neuron-like unit l of an artificial neural network is generally specified by a nonlinear function $g(u_l)$ of its input u_l . In turn, the input to unit l is generally taken to be a linear superposition

$$u_l = \sum_{l'} w_{ll'} a_{l'} + w_{l0} \quad (15)$$

of the activities $a_{l'}$ of the units l' that extend connections to it, the coefficients in the linear combination being the weights $w_{ll'}$ of these connections. Additionally, a unit l is assigned a bias w_{l0} that may be regarded as the weight of a connection to l from a fictitious unit 0 with perpetual activity $a_0 \equiv 1$. The nonlinear function $g(u)$ that transfers the single-unit input into a corresponding output is called a “squashing function,” since it squashes an input u on the real line down to a value on a restricted interval, usually $[0, 1]$ or $[-1, 1]$. In the standard three-layer feedforward net with input neurons k , hidden neurons j , and output neurons i , we would have $a_i = g(\sum_j w_{ij} a_j)$ and $a_j = g(\sum_k w_{jk} a_k)$. Since the input units serve merely as registers for the $\mathbf{x}$ variable, the only network parameters open

to modification during learning are the weights of connections into hidden or output units (and the biases of these units). The label i may be suppressed in a network with a single output unit, as considered here. The indices l and l' are reserved for “generic” units.

In imitation of standard backprop, the noise-adapted learning algorithm that we propose and demonstrate is based on a gradient-descent approach to minimization of the cost function in weight space. We also follow common practice [13] in departing from strict gradient descent by updating weights after each pattern presentation. In a given pass through the training set (a given “epoch”), the patterns are presented once each in random order. (Strict gradient descent would require that the weights be updated after each presentation of the full set of patterns – which is called “epochal” or “batch” updating.) Thus we implement an “on-line” or “stochastic” updating scheme, which has certain advantages [1,18] over the “batch” procedure, notably the possibility of escaping from local minima of E in weight space.

The weights are initialized at a randomly chosen point in weight space, and a learning rate η is selected. We propose and test the following training routine:

- (a) Input a randomly chosen pattern r , and calculate the corresponding activity $a^{(r)}$ of the output unit.
- (b) Update the modifiable weights (including biases) via

$$\Delta w_{ll'}^{(r)} = -\eta \frac{\partial E_r}{\partial w_{ll'}} , \quad (16)$$

while keeping σ at its current value.

- (c) Repeat (a)-(b) until all patterns have been presented without repetition, completing an epoch.
- (d) *Keeping the weights at their current values*, solve Eq. (13) for σ by iteration, with the conventional rms error measure $\sigma_m^{[0]}$ of Eq. (10) as input (a few iterations will ordinarily suffice for convergence).
- (e) Repeat (a)-(d) for as many epochs as required to achieve convergence of the learning process, with neither σ nor E showing significant further decrease.

We note that the rule (15) can be cast into the same form

$$\Delta w_{ll'}^{(r)} = -\tilde{\eta} \frac{\partial E_r^{(o)}}{\partial w_{ll'}} \quad (17)$$

as the weight correction applied in ordinary stochastic backprop if we use a modified or “effective” learning rate

$$\tilde{\eta} = \eta / \left[1 + \left(\frac{\sigma_r}{\sigma} \right)^2 \right], \quad (18)$$

where η is the “bare” learning-rate parameter fixed at a suitable value $0 < \eta < 1$.

Referring to Eqs. (17) and (18), several interpretive comments on the proposed learning algorithm may now be offered.

- (i) If the standard deviation σ_r assigned to the data point r is large compared to the current value of the model error measure σ , then the change in weights is small. In other words, experimental data with large error bars are not given much weight in the learning (fitting) process – which is just the qualitative behavior we seek.
- (ii) Developing further on this behavior, assume that, in the course of the training procedure, σ has become very small compared to all the σ_r (as when a very good model has been created). Then the denominator $1 + (\sigma_r/\sigma)^2$ in relation (18) will be very large for all training patterns, the effective learning rate $\tilde{\eta}$ will approach zero, and significant weight changes cease to occur. Thus, the new algorithm has the salutary property that training fades out as the model improves and the estimated error σ becomes negligible. In the limiting case $\sigma \rightarrow 0$, this inherent weight-change decay prevents the “perfect” neural-network theory then achieved from being destroyed in further training cycles. By contrast, in ordinary backprop the quality of a network model may deteriorate with prolonged training as the system learns the spurious features of the noisy data.
- (iii) What if the experimental standard deviation is the same for all patterns? Then the denominator on the right in Eq. (18) is just some pattern-independent factor that adjusts the learning rate $\tilde{\eta}$ depending on the value of σ for the current epoch.

Within the limitations of its parametric resources, the net will learn the given data indiscriminately, errors and all, as in standard backprop. However, if η is held fixed during the learning process, it is still true that weight modifications are quenched as the model improves.

- (iv) In the limiting case that the experimental errors are all negligible compared to σ (as in the extreme case of a bad model of the results of good experiments, or perhaps in the early phases of training), the learning algorithm reduces to that of vanilla backprop, since $\tilde{\eta}$ becomes identical with η in Eq. (18). This reduction is foretold at an earlier point in the development: with $\sigma_r \ll \sigma$, $r = 1, \dots, n$, the usual rms expression (10) for σ may be retrieved from Eq. (6).
- (v) Assumption **(I)** may fail in the initial stages of learning, when systematic errors (though attributable to the random starting weights) may dominate. But with large model errors, the effect of experimental errors can be ignored and the procedure reduces to standard backprop.

As already indicated in (ii) and (iii) above, the proviso that the modified learning procedure be implemented at fixed η has the interesting consequence that, given a steady improvement in the model epoch by epoch, there will be a corresponding staged decline in the effective learning rate $\tilde{\eta}$. Staged reductions in learning rate have proven beneficial to the creation of high-quality models in problem contexts where the data is regarded as noise-free (for example, see [7,8]).

The value of σ for the *predictive mode* of network performance is obtained by solving the nonlinear equation (13) for the case that the sum on r runs over the *test* set, with the weights taken at their values for the mature network.

The above analysis and procedure have straightforward generalizations. The extension to the case of several dependent or output variables Y_k , $k = 1, \dots, K$, each a function of $\mathbf{x}$, is transparent. A corresponding neural network model would most naturally contain an output unit to represent the result of the network’s computation for each such variable

Y_k . (We are reminded of the theorem of Hornik, Stinchcombe, and White [19] establishing that any Borel-measurable function from one finite-dimensional space to another finite-dimensional space can be approximated to any prescribed accuracy by a three-layer neural network (having one hidden layer and arbitrary squashing function).) The treatment is also readily adapted to the case that the model result for a given dependent variable is encoded in the activities of more than one output unit.

As is sometimes indicated by the nature of the problem [20], we may also relax the assumption (I) that the model values for the quantity Y differ from those generated with the true mapping by an additive normally distributed random number with *zero* mean; instead we may take $\epsilon_m^{(r)} \in N(\mu_m, \sigma_m^2)$ in Eq. (2). Eqs. (3)-(9) are generalized to arbitrary mean error μ_m by the replacement $y_m^{(r)} \rightarrow y_m^{(r)} + \mu_m$. In the new version of (5), the differentiation with respect to the fitting parameters ω_p is to be performed with *both* σ_m and μ_m held constant, and Eq. (6), which determines the model standard deviation σ_m , is to be supplemented by a second equation

$$\sum_{r=1}^n \frac{y_e^{(r)} - y_m^{(r)} - \mu_m}{(\sigma_e^{(r)})^2 + \sigma_m^2} = 0, \quad (19)$$

determining the mean model error μ_m . Corresponding replacements are to be made in the modified backpropagation algorithm; thus, after each epoch, the estimates of both σ and $\mu = \mu_m$ are to be updated via self-consistent iterative solution of

$$\sigma^2 = \sum_{r=1}^n c_r [(t^{(r)} - a^{(r)} - \mu)^2 - \sigma_r^2], \quad (20)$$

and

$$\mu = \sum_{r=1}^n c_r (t^{(r)} - a^{(r)}), \quad (21)$$

with the c_r again given by (14).

3. Numerical Applications

We now consider two specific applications of the modified backpropagation algorithm introduced in section 2. The results will serve to illustrate the effectiveness of the new “cleanprop” learning scheme in suppressing “experimental” errors when training neural networks on noisy data.

3.1. Application 1: Liquid-drop model of atomic masses

The first application, although rather academic in nature, is the more informative as a test of the method proposed, in that it allows comparison of the output $y_m(\mathbf{x}^{(r)}) = y_m^{(r)}$ of the neural-network model for a given pattern r , with the *true* value $Y(\mathbf{x}^{(r)}) = Y^{(r)}$ of the property Y . The true values of the desired quantity are generally unknown in real-world applications, such as the second trial case to be discussed below, since they are usually obscured by noise or experimental error. To examine instead a situation that is fully under our control, we choose some well-defined function $Y(\mathbf{x})$ and generate a collection of pairings $(\mathbf{x}^{(r)}, Y^{(r)})$ for a list of inputs $\mathbf{x}^{(r)}$. These pairings – or “patterns” – constitute the true data. The true values $Y^{(r)}$ are then artificially contaminated with white noise, the “experimental” error ϵ_r for each instance r being randomly drawn from a Gaussian distribution with zero mean and prescribed standard deviation or error bar σ_r . Thus the “experimental” values of Y are given by $y_e^{(r)} = Y^{(r)} + \epsilon_r$. From the set of noisy examples $(\mathbf{x}^{(r)}, y_e^{(r)})$ thus assembled, some are selected at random to create a test set that is used to assess the predictive power of trained network models. The remaining patterns r form the training set, which furnishes nominal targets $t^{(r)} = y_e^{(r)}$ for the learning process. By construction, the test set and training set have the same statistical properties, provided both are sufficiently large.

To proceed from the abstract to the concrete, we must of course specify the “true” map $\mathbf{x} \rightarrow Y(\mathbf{x})$ and the distribution of error bars σ_r . The exercise will be more illuminating if the problem addressed is not purely academic. Accordingly, we have chosen a map

that corresponds to a schematic version of a real-world problem that has been studied in earlier work with the standard backpropagation algorithm [6,7,21]. Multilayer feedforward neural networks have been taught to model the systematics of the atomic mass table with some precision. In this application, the input variable $\mathbf{x}$ has as its components the proton number Z and the neutron number N of a given nuclide, or nucleus, while the dependent variable Y is the mass excess

$$\Delta m(Z, N) = m(Z, N) - Am_u \quad (22)$$

of that nuclide. The mass excess is the difference between the mass $m(Z, N)$ of the neutral atom and a reference mass given by the number of nucleons in the nucleus, $A = N + Z$, times the atomic mass unit $m_u = 931.48$ MeV, which is one-twelfth the mass of a neutral ^{12}C atom. Smaller mass excess corresponds to stronger binding (overwhelmingly nuclear in origin) relative to the standard set by ^{12}C .

In the earlier work, the neural network systems were trained on masses that were (mostly) determined by experiment and thus include complex effects due to like-nucleon pairing and nuclear shell structure. Here we consider the much simpler problem in which the “true” masses are generated not by nature but instead from the familiar Bethe-Weizsäcker semi-empirical mass formula [22]

$$m(Z, N) \doteq Zm_H + Nm_n - a_1A + a_2A^{2/3} + a_3Z^2/A^{1/3} + a_4(Z - N)^2/A, \quad (23)$$

wherein m_H and m_n are the masses of the hydrogen atom and the neutron. The last four terms in (23) are motivated by a liquid-drop model of the nucleus and represent, in order, the volume, surface, Coulomb, and (a)symmetry contributions to the mass. We adopt the coefficients $a_1 = 15.68$ MeV, $a_2 = 18.56$ MeV, $a_3 = 0.717$ MeV, and $a_4 = 28.1$ MeV determined empirically by Myers and Swiatecki [23]. For our initial exploration of the cleanprop algorithm, the “true” values $Y(\mathbf{x})$ are identified with the mass excesses of the liquid-drop model and accordingly are specified by Eq. (22) with $m(Z, N)$ calculated from

Eq. (23). It should be noted that we disregard the pairing correction and more complicated shell corrections that are sometimes attached to the liquid-drop formula.

The options for specifying the distribution of standard deviations σ_r are quite open. In a typical application they would be provided (by the natural or computational experimentalist) along with the nominal experimental values $y_e^{(r)}$. In view of the physical interpretation of the test problem we have adopted, it would seem most reasonable to chose a distribution of σ_r values such that larger error bars are assigned to nuclides in the (Z, N) plane lying farther from the line of stable nuclei. However, for our purposes it is clearly more instructive to assume a generic distribution rather than one that is so context-specific. Thus we have chosen to assign error bars σ_r to the individual “measurements” r by sampling a Gaussian distribution with zero mean and standard deviation Γ , i.e., $\sigma_r \in N(0, \Gamma^2)$. The behavior of the cleanprop algorithm will be studied as Γ is increased in steps to larger and larger values and the noise tends to swamp the signal.

As already remarked, this test problem has the advantage (over real-world problems) of a known mathematical relationship between the input variables and the desired dependent variable, and a known mechanism for masking this dependent variable with “experimental” uncertainties. Indeed, the key assumption **(II)** made in section 2 – which is not necessarily valid in applications to real-world data – holds by construction. On the other hand, the validity of key assumption **(I)** is still subject to question in the present application, since it depends on unknown aspects of neural-network modeling. Numerical evidence will be offered that **(I)** does in fact hold approximately for the models of the liquid-drop problem that we develop.

The computational study proceeds with “true” mass excess values and associated errors generated by the formula (23) for 2251 different nuclides (drawn from the 2291 used in most of the earlier neural network studies of the atomic mass table [6,7,21,24]). These nuclei lie in or relatively near the valley of stability and span most of the domain of stable and unstable nuclides observed in nature. All pairs of learning/test runs for the liquid-drop

problem are based on the same training and test sets. The training set is composed of 1700 randomly selected nuclides, leaving 551 for testing. Figure 1 shows the locations, in the (Z, N) plane, of the nuclides in these training and test sets.

The neural networks that are taught to model the systematics of the liquid-drop masses have a layered, feedforward architecture. More specifically, all forward connections between nodes in successive layers are present, but no backward or lateral connections are permitted, and connections do not skip layers. The input layer contains two units, whose activities encode the values of Z and N , while the output layer consists of a single unit displaying a coded value for the corresponding mass excess. In addition to the input and output layers, the networks possess two or three intervening hidden layers. All units have graded activities a_l lying in the interval $[0,1]$. For the squashing function we make the conventional choice

$$a_l = g(u_l) = \frac{1}{e^{-u_l} + 1} . \quad (24)$$

Analog representations are employed at input and output layers. Scaling factors of $1/120$ and $1/140$ are applied to fit the relevant Z and N values into the activity range $[0,1]$ of the input units. (Although input-unit activities need not be subject to squashing since they only record the input patterns for subsequent analysis, it has been argued [25] that such preprocessing is desirable.) Similarly, an appropriate scaling factor and its inverse are used to translate the target mass excess (including experimental error) to a target activity on $[0,1]$ and to convert the actual activity of the output unit to a mass excess in MeV.

We shall not be concerned with achieving very precise fits of the mapping defined by the liquid-drop model, since our primary aim is to establish the utility of the new learning procedure in reducing the effect of target noise in supervised training of neural networks. Nevertheless, it is well to demonstrate that highly accurate modeling of the liquid-drop data set is possible in the noise-free case. The results of some training runs based on standard backprop are collected in table 1. (Here, and throughout, we make use of the notation for network types introduced in ref. [6]. For example, $^a(2 + 5 + 5 + 5 + 1)_a[81]$ is

the designation for an exclusively feedforward, no-skip network having 2 input units, three successive hidden layers of 5 units each, and one output unit, with analog coding schemes at both input and output interfaces and 81 modifiable weight/bias parameters.)

One very positive feature may be inferred from the results for the rms model error seen in table 1: all eight of the sample models perform essentially as well on the test set as they do on the training set. This is often taken as an indication of good generalization, although the more conservative statement is that all the features learned by such networks are common features of training and test sets. Clearly, some benchmark is needed to judge the accuracy of these models. The scale of the $\sigma^{[o]}$ error estimates in table 1 can be compared with the mass excess values themselves, which range from about -90 MeV up to some 170 MeV. Even the simplest of the four network types considered, $^a(2 + 3 + 3 + 1)_a[37]$, is capable of reproducing the test and training data with a typical relative error below 3%. It may further be helpful to remark that a conventional mass model fitting the experimental masses with a $\sigma^{[o]}$ value of 0.2 MeV would be regarded as highly competitive, even with ~ 400 adjustable parameters [26]. Most popular among the conventional fits of the empirical data are the macroscopic-microscopic models of Möller and Nix [15] (favored because of their high theoretical input and small number of free parameters). The latest versions have σ values around 0.8 MeV, obtained with about 30 adjustable parameters [20]. On the other hand, for sets of training and test nuclides very similar to those employed here, the most refined neural network models of the Brookhaven atomic mass table [7,24] have $\sigma^{[o]}$ values around 0.6 MeV for learning and around 0.7 MeV for prediction, with some 350 modifiable weights. It is true that the realistic mass problem – with a rougher mass surface – is more difficult than the liquid-drop problem. On the other hand, significant improvements on the performance shown in table 1 are undoubtedly available through increases in the number of hidden units; through such elaborations upon backpropagation as scheduled reduction of the learning rate, more frequent presentation of hard-to-learn patterns, and pruning of unimportant connections followed by retraining

[7]; and/or through other strategies including conjugate-gradient search, skip-connections, and singular-value preprocessing [24].

We now turn to details of the implementation of the modified backpropagation procedure. In fact, the new procedure is extremely convenient to apply since, between σ updates via Eq. (13), it operates just like ordinary backprop except for the presence of pattern-specific learning rates (18). Calculating the derivative in Eq. (17), the prescription for updating weights and biases upon presentation of pattern r takes the more explicit form

$$\Delta w_{ll'}^{(r)} = \eta \left[1 + \left(\frac{\sigma_r}{\sigma} \right)^2 \right]^{-1} \delta_l^{(r)} a_{l'}^{(r)} + \alpha \Delta w_{ll'}^{(r-1)} \quad (25)$$

in terms of error signals $\delta_l^{(r)}$, where l is a hidden or output unit and l' is a hidden or input unit. Here we have supplemented the rule (17) with a so-called momentum term [13] to suppress oscillations in weight space that might otherwise occur during learning. The momentum parameter has been set to $\alpha = 0.9$ in all numerical applications. In Eq. (25) the index $r-1$ stands for the pattern presented immediately prior to the current pattern r . When updating a bias w_{lo} , the activity $a_o^{(r)}$ appearing in the corresponding version of (25) is trivially unity. For the output unit the error signal is

$$\delta_j^{(r)} = (t^{(r)} - a^{(r)}) g'(u^{(r)}) = a^{(r)} (1 - a^{(r)}) (t^{(r)} - a^{(r)}) , \quad (26)$$

where $a^{(r)}$ and $u^{(r)}$ denote the activity of the output unit and the net stimulus it receives upon presentation of pattern r at the input interface. For hidden units j the error signals are given by

$$\delta_j^{(r)} = \sum_{l'} \delta_{l'}^{(r)} w_{lj} g'(u_j^{(r)}) = a_j^{(r)} (1 - a_j^{(r)}) \sum_{l'} \delta_{l'}^{(r)} w_{lj} . \quad (27)$$

Weights and biases are adjusted according to Eqs. (25)–(27) before the next pattern is imposed.

The initial weights and biases are drawn from a uniform distribution in the interval $[-0.5, 0.5]$. For each epoch of training, the order of presentation of patterns is refreshed by random permutation. Thus the training process follows a stochastic path in pattern space.

In conformity with the training strategy (a)–(e) specified in section 2, the quality figure σ is readjusted after each epoch. Like the rms error (10) in ordinary stochastic backprop, the error measure σ determined by Eq. (13) exhibits oscillations when plotted against the number of completed epochs. Since we wish to find the best available network, our training code includes an instruction to store the point in weight space corresponding to the minimal error σ recorded over the pre-set number of learning epochs. The network models discussed below were found with this procedure. (The nets of table 1 were obtained via the analogous optimal stopping prescription within ordinary backprop.) To facilitate a simpler assessment of the efficacy of the cleanprop algorithm, we shall forgo such refinements of the training programs as pruning, more frequent exposure to “difficult” examples, etc.

We have chosen the $^a(2 + 3 + 3 + 1)_a$ [37] network class for thorough study. This model class admits satisfactory accuracy in both learning and prediction, while allowing a large number of training experiments to be carried out in a relatively short time. A representative selection of results from systematic cleanprop runs on noise-corrupted liquid-drop data is presented in tables 2–4. Each table refers to a different value of $\mathcal{W} = \Gamma\sqrt{2\ln 2}$, the halfwidth at half maximum of the normal distribution $N(0, \Gamma^2)$ that is sampled to produce the “experimental” error bars σ_r . The quality of individual networks, in learning or prediction, is assessed in terms of three performance figures, $\sigma^{[o]}$, σ , and $\sigma^{[tr]}$. The first two are the usual rms error measure $\sigma_m^{[o]} \equiv \sigma^{[o]}$ defined by Eq. (10) and the theoretical error estimate σ calculated from Eq. (8) or Eq. (13). Since, in the present application, the actual values of the desired quantity – the mass excess in the liquid-drop model – are known, we are also in a position to form the rms deviation of the masses generated by the model from their respective *true* (as opposed to experimental) values,

$$\sigma^{[tr]} = \left[\frac{1}{n} \sum_{r=1}^n (Y^{(r)} - y_m^{(r)})^2 \right]^{1/2} . \quad (28)$$

This is the third performance measure entered in the tables. It provides the most telling indication of how well a network is modeling the actual mapping. All three figures of merit

are reported in MeV (as is conventional practice in mass modeling) and hence include the factor that scales the network output to its corresponding mass-excess value. Different pairs of training/test entries in a given table belong to different choices of (a) the set of random choices of the “experimental” errors ϵ_r from the distributions $N(0, \sigma_r^2)$ and/or (b) the starting point in weight space and the stochastic training path in pattern space. All entries in a given table correspond to the same choice of σ_r values. To provide baselines for judging the quality of the network models developed using the cleanprop procedure, tables 2-4 contain some reference entries for nets trained by ordinary backpropagation, either on noise-free data or on standardized noisy data. Each of these backprop nets shares the same starting point/stochastic path and (if applicable) the same noisy data with one of the cleanprop nets. (It should be noted that the sets of starting points/stochastic paths used for tables 2, 3, and 4 are not identical, although they are given the same labels $i_t = 1, 2, 3, (4)$. The sets 1, 2, 3 used for the $\mathcal{W} = 15$ and 25 MeV cases of table 4 do coincide, but the output coding schemes applied at the two noise levels differ somewhat.)

A number of useful conclusions may be drawn from these tables.

- ◊ The performance measure $\sigma^{[o]}$ is typically comparable to (or somewhat larger than) the standard deviation $\Gamma \approx 0.85 \mathcal{W}$ of the normal distribution from which the error bars σ_r were chosen. This is expected, since the networks in question have far too few adjustable parameters to fit the given data (37 weights and thresholds to reproduce data generated by 1700 random variables). With a substantial increase in the number of processing units, feedforward network models (trained e.g. by backprop) could precisely fit the noisy data – but this is *not* desirable, since predictive performance would probably suffer.
- ◊ That cleanprop is effective in screening out the noise is documented by the fact that both σ and $\sigma^{[tr]}$ are closely comparable in size and are in most cases considerably smaller than both $\sigma^{[o]}$ and Γ . The latter feature persists over the range of noise levels $\mathcal{W}$ involved, even out to $\mathcal{W} = 25$ MeV, which corresponds to very heavy contami-

nation. The only overt sign of deterioration of the cleanprop models with increasing noise is a growing discrepancy between σ and $\sigma^{[tr]}$ (evident in table 4).

- ◊ At the larger noise levels, the cleanprop networks have much better performance figures than nets trained by backprop on the same data. Even so, ordinary backpropagation does show some ability to filter out noise. This ability may be attributed – in part – to the smoothness of the semi-empirical mass formula.
- ◊ Examining the results for $\sigma^{[tr]}$, it is seen that some of the models obtained via cleanprop are reasonably good on an absolute scale, even when the data used to train them is extremely noisy. Cases in point are provided by the 1(15)1, 3(15)1, 3(15)3, 1(25)1, and 1(25)3 cp entries in table 4. Moreover, there are examples in the tables to demonstrate that cleanprop is capable of processing corrupted data to yield network models with $\sigma^{[o]}$ values comparable to those of networks trained with backprop on noise-free data, using the same starting weights and training path in pattern space.
- ◊ The quality of models depends sensitively on the starting point in weight space and/or the stochastic path in pattern space. For given $\mathcal{W}$, the quality of solution is generally not very sensitive to the specific choice of experimental errors ϵ_r , although there are some examples in the tables in which this choice does significantly alter the learning trajectory in weight space. (In table 2, compare the 3(5)1 entry with the 1(5)1 and 2(5)1 entries and the 3(5)3 entry with the 1(5)3 and 2(5)3 entries; and, in table 4, the 3(15)3 case with the 1(15)3 and 2(15)3 results.) Variations of model performance on shifting from one noise level to another are quite visible. However, as attested by further results not shown, such variations are typically rather mild if the starting point and stochastic pattern path are left unchanged. On general grounds, the experimental errors should of course have a greater influence on the final network configuration as $\mathcal{W}$ is increased, assuming that the learning process is not quenched.
- ◊ Predictive accuracy is comparable to that of learning. The nets use their limited resources to learn features that are common to training and test sets. That performance

on test and training sets is similar is a good sign that the network is seeing through the noise and attempting to fit the underlying function. Satisfactory generalization is facilitated by the smoothness of the function to be learned, which is believed to be responsible for the fact that in many cases $\sigma^{[0]}$ is slightly smaller for the test set than for the training set.

The basic conclusion is that the cleanprop algorithm works very well in this application. Even when the data is contaminated with large errors, the modified backpropagation scheme can yield good approximations to the true mapping. A comparative study of the behavior of the method at extreme values of $\mathcal{W}$, namely 25, 50, and 75 MeV, indicates that failure of the algorithm with increasing noise level is extraordinarily graceful. (This study examines sets of cases in which the initial weights and pattern path are kept the same, while a fixed distribution of the σ_r and a fixed distribution of the ϵ_r is appropriately scaled with $\mathcal{W}$.) To some extent, such robustness must depend on the very smooth character of the target mapping.

Whatever the problem, the cleanprop procedure must ultimately cease to be useful in the high-noise regime, if for no other reason than the following. If all σ_r become very large, the effective learning rate will become very small even for moderately large σ . Thus the learning process will stagnate before a good model has been achieved.

To complete the picture for this application, we have made checks of the validity of assumption **(I)** underlying the maximum likelihood treatment of section 2 that led to the cleanprop scheme. For nets trained with cleanprop on the noise-corrupted liquid-drop data, the distribution of model errors $\epsilon_r = \epsilon_m^{(r)}$ is found to conform approximately with the corresponding Gaussian $N_m(0, \sigma_m^2)$. An example is shown in fig. 2.

3.2. Application 2: Modeling experimental mass systematics

Continuing the theme of mass modeling, it is natural to perform an analysis on the *actual* experimental nuclidic mass data, consisting of 1654 mass excess values $y_e^{(r)} = \Delta m_e^{(r)}(Z^{(r)}, N^{(r)})$ together with their assigned error bars $\sigma_e^{(r)}$. Evidently, in this prob-

lem we no longer have the advantage of knowing the true values $Y^{(r)}$ of the mass excess. On the other hand, the data now reflects the discontinuous character of the realistic mass surface $m(Z, N)$, which is attributable primarily to pairing and shell effects, and also contains apparent irregularities that are yet to be explained in physical terms.

The cleanprop learning algorithm has been seen to function remarkably well in filtering out random errors superimposed on the semi-empirical mass formula (23). However, it is expected to be less effective when the underlying function to be approximated is jagged or erratic instead of smooth. Our second example serves to test the procedure on a problem that contains far more complex intrinsic structure, although the test is not a very stringent one since the experimental uncertainties are not large in this case. (The average error bar, over the full data set, is only $\bar{\sigma}_r = 0.0412$ MeV. The largest uncertainties, reaching 1.5 MeV, occur on the boundaries of the domain of observed nuclei.)

For the purposes of our study, the full data base is divided into a training set of 1323 nuclides, selected at random, and the complementary test set of 331 examples. In choices of architecture and coding, we follow the lead of earlier phenomenological modeling of the experimental atomic mass table in terms of feedforward neural networks [6,7]. Accordingly, attention is restricted to networks of the type ${}^h(18 + 10 + 10 + 10 + 1)_a[421]$, which, on implementing various refinements of standard backpropagation training, has been shown to admit precise modeling of the raw data. The input coding scheme involves two sets of eight “on-off” neurons representing Z and N in binary, plus two analog neurons that (redundantly) encode scaled values of A and $N - Z$. Specifications are otherwise essentially the same as in the simpler problem of capturing the liquid-drop formula, and application of the cleanprop training rule proceeds in a similar manner.

Results obtained with the actual data are presented in table 5. The rms error measure $\sigma^{[0]}$ and the derived theoretical error figure σ , calculated for training and test sets, are given at specified stages for two networks developed from different starting points in weight space through different paths in pattern space. Since the experimental mass data is of high

quality, the improvement gained by cleanprop is expected to be marginal, and the difference $\sigma^{[o]} - \sigma$ is indeed found to be very small. In passing, we note that good generalization ability is also exhibited by these more elaborate mass models.

4. Conclusions and Prospects

A modified backpropagation procedure has been developed for training layered feed-forward neural networks on data points r with noisy values of the target variables, following the dictum that the noisier examples, with larger error bars σ_r , should be accorded less influence in the development of the connection weights that define the network model. This dictum has been realized quantitatively through application of a maximum-likelihood criterion, which leads to the introduction of a new cost function to replace the mean-square error and a procedure for estimating a new measure σ of the model error. The estimated model error provides a more realistic indication of the accuracy of the network's approximation to the true, uncontaminated mapping than does the conventional rms error $\sigma^{[o]}$ with respect to the noisy targets. The new training scheme is easy to implement, in that it departs from standard on-line backpropagation only in (a) the introduction of a pattern-specific learning rate $\eta/[1 + (\sigma_r/\sigma)^2]$ and (b) iterative re-determination of the model error estimate σ after each epoch. The maximum-likelihood treatment assumes that the errors made by the model, relative to the true mapping, are normally distributed with zero mean and standard deviation σ , and that the errors in the targets corresponding to given instances r of the map (the "experimental errors") are normally distributed with zero mean and standard deviations σ_r . As necessary, these assumptions can be relaxed and the approach extended to problems with more complicated or less orthodox statistical properties.

In specific applications to the modeling of atomic masses, the new training algorithm (called "cleanprop") has proven to be very effective in suppressing the harmful effects of noise on the modeling process. Computational evidence has been presented which demonstrates that networks trained with cleanprop are capable of building satisfactory models of the true mapping even at high noise levels, provided the function to be approximated is reasonably smooth. It is anticipated that the cleanprop algorithm will be useful in a wide range of neural network applications in which a statistical characterization of database errors is available.

Acknowledgments

This research has been supported in part by the U. S. National Science Foundation under Grant No. PHY-9307484 and by the BASF Aktiengesellschaft and the Studienstiftung des deutschen Volkes through a postdoctoral fellowship award to KAG. We are grateful to P. Möller and J. R. Nix for very helpful communications. It was their earlier work on the use of experimental mass uncertainties in determining model parameters that stimulated this investigation. JWC has enjoyed the hospitality of the Theoretical Physics Institute of the University of Coimbra and the Physics Division of Argonne National Laboratory during the completion of this article. One of the authors (K.A.G.) would like to thank Professor Abdus Salam, the International Atomic Energy Agency and UNESCO for hospitality at the International Centre for Theoretical Physics, Trieste.

References

- [1] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Redwood City, CA, 1991).
- [2] J. W. Clark, *Phys. Med. Biol.* 36 (1992) 1259.
- [3] N. Qian and T. J. Sejnowski, *J. Mol. Biol.* 202 (1988) 865; H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, B. Lautrup, L. Nøskov, O. H. Olsen, and S. B. Petersen, *FEBS Letters* 241 (1988) 223; H. Bohr, J. Bohr, S. Brunak, R. M. J. Cotterill, H. Fredholm, B. Lautrup, and S. B. Petersen, *FEBS Letters* 261 (1990) 43; G. L. Wilcox, M. Poliac, and M. N. Liebman, *Tetrahedron Comp. Meth.* 3 (1990) 191; P. Stolorz, A. Lapedes, and Y. Xia, *J. Molec. Biol.* 225 (1991) 363.
- [4] J. R. P. Angel, P. Wizinowich, M. Lloyd-Hart, and D. Sandler, *Nature* 348 (1990) 221; D. G. Sandler, T. K. Barrett, D. A. Palmer, R. Q. Fugate, and W. J. Wild, *Nature* **351**, 300 (1991); M. Lloyd-Hart, P. Wizinowich, B. McLeod, D. Wittman, D. Colucci, R. Dekany, D. McCarthy, J. R. P. Angel, and D. Sandler, *Ap. J.* 390 (1992) L41.
- [5] B. Denby, *Comput. Phys. Commun.* 49 (1988) 429; C. Peterson, *Nucl. Instr. Methods* A279 (1989) 537; L. Lönnblad, C. Peterson, and T. Rönngvaldsson, *Comp. Phys. Commun.* 70 (1992) 167; C. Bortolotto, A. de Angelis, N. de Groot, and J. Seixas, *Int. J. Mod. Phys. C3* (1992) 733.
- [6] S. Gazula, J. W. Clark, and H. Bohr, *Nucl. Phys.* A540 (1992) 1.
- [7] K. A. Gernoth, J. W. Clark, J. S. Prater, and H. Bohr, *Phys. Lett.* 300 (1993) 1.
- [8] K. A. Gernoth and J. W. Clark, *Neural Networks*, in press.
- [9] J. W. Clark, K. A. Gernoth, and M. L. Ristig, in: *Condensed Matter Theories*, vol. 9, eds. J. W. Clark, A. Sadiq, and K. A. Shoaib (Nova Science Publishers, Commack, NY, 1994).
- [10] E. A. Wan, *IEEE Trans. Neural Networks* 1 (1990) 303.
- [11] M. Oppner and D. Haussler, *Phys. Rev. Lett.* 66 (1991) 2677.
- [12] D. J. C. MacKay, *Neural Computation* 4 (1992) 415; 4 (1992) 448.

- [13] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vols. 1 and 2 (MIT Press, Cambridge, MA, 1986).
- [14] H. Cramér, *Mathematical Methods of Statistics* (Princeton University Press, Princeton, NJ, 1946).
- [15] P. Möller and J. R. Nix, *At. Data Nucl. Data Tables* **39** (1988) 213.
- [16] R. von Mises, *Mathematical Theory of Probability and Statistics* (Academic Press, New York, 1964).
- [17] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).
- [18] E. Barnard, *IEEE Trans. Neural Networks* **3** (1992) 232.
- [19] K. Hornik, M. Stinchcombe, and H. White, *Neural Networks* **2** (1989) 359.
- [20] P. Möller and J. R. Nix in: *Nuclei Far From Stability/Atomic Masses and Fundamental Constants 1992* (Bernkastel-Kues), IOP Conference Series No. 132, eds. R. Neugart and A. Wöhr (Institute of Physics Publishers, Bristol, 1993), pp. 43-52.
- [21] J. W. Clark, S. Gazula, K. A. Gernoth, J. Hasenbein, J. Prater, and H. Bohr, in: *Recent Progress in Many-Body Theories*, vol. 3, eds. T. L. Ainsworth, C. E. Campbell, B. E. Clements, and E. Krotscheck (Plenum, New York, 1992), p. 371.
- [22] A. Bohr and B. R. Mottelson, *Nuclear Structure*, vol. 1 (W. A. Benjamin, New York, 1969).
- [23] W. D. Myers and W. J. Swiatecki, *Nucl. Phys.* **81** (1966) 1.
- [24] B. L. Kalman, private communication.
- [25] S. M. Weiss and C. A. Kulikowski, *Computer Systems that Learn* (Morgan Kaufmann, San Mateo, CA, 1990).
- [26] P. E. Haustein, *At. Data Nucl. Data Tables* **39** (1988) 185.

Table 1

Performance of neural network models on the liquid-drop mass problem for noiseless data. Performance is assessed by the conventional rms error measure $\sigma^{[o]}$ of Eq. (10), quoted in MeV. The training set consists of 1700 patterns (nuclides (Z, N) together with their associated mass excess values of Eqs. (22)–(23)) selected at random from the full set of 2251. The remaining 551 patterns serve as test examples. The number of epochs of training is denoted $N(\text{epochs})$. The three entries for the first network architecture correspond to different sets of initial weights, whereas the same set of initial weights is used in all runs for the second architecture.

Net type	N(epochs)	$\sigma^{[o]}(\text{train})$	$\sigma^{[o]}(\text{test})$
$^a(2 + 3 + 3 + 3 + 1)_a[37]$	20,000	1.336	1.216
"	20,000	3.942	3.992
"	20,000	4.436	4.335
$^a(2 + 5 + 5 + 5 + 1)_a[81]$	3000	0.518	0.602
"	10,000	0.295	0.301
"	20,000	0.245	0.232
$^a(2 + 7 + 7 + 1)_a[85]$	5000	0.630	0.622
$^a(2 + 9 + 9 + 1)_a[127]$	3000	0.602	0.580

Table 2

Test of the modified backpropagation procedure (“cleanprop”) for training multilayer feed-forward nets on noisy data. The quoted results refer to networks of type $^a(2 + 3 + 3 + 3 + 1)_a$ [37]. For all training runs, the prescribed maximum number of epochs was 20,000, the learning rate 0.5, and the momentum parameter 0.9. The quantity $\sigma^{[o]}$ is the root-mean-square error calculated on the basis of the noisy data, $\sigma = \sigma_m$ is the model or theoretical error, and $\sigma^{[tr]}$ is the root-mean-square error with respect to the true, noise-free mass-excess values provided by the liquid-drop model; all three measures are quoted in MeV. The training set consists of 1700 and the test set of 551 patterns. The “experimental” error ϵ_r contained in the target output value for pattern r is chosen at random from a pool of 400 values normally distributed with zero mean and standard deviation σ_r . The experimental standard deviation for pattern r was itself chosen randomly from a normal distribution with zero mean and halfwidth $W = 5$ MeV. The notation $i_e(W)i_i$ is used to distinguish the different paired training/test runs. The index i_e designates a specific choice of experimental errors ϵ_r , while i_i labels a specific choice of initial weights and stochastic training path in pattern space. Noise-free training/test runs are denoted $\bullet(0)i_i$. For each pair of runs, the first [second] row gives the error measures evaluated for the training [test] set. The training rule implemented is either cleanprop (cp) or ordinary backprop (bp).

Run	Rule	Mode	$\sigma^{[o]}$	σ	$\sigma^{[tr]}$
1(5)1	cp	train	6.615	4.990	5.074
"	—	test	6.196	4.072	4.436
•(0)1	bp	train	5.031	5.031	5.031
"	—	test	4.415	4.415	4.415
1(5)2	cp	train	4.762	1.847	2.037
"	—	test	4.609	1.582	1.891
1(5)2	bp	train	4.641	1.780	1.885
"	—	test	4.608	1.988	1.995
•(0)2	bp	train	2.285	2.285	2.285
"	—	test	2.105	2.105	2.105
1(5)3	cp	train	6.316	4.575	4.709
"	—	test	6.430	4.521	4.645
1(5)3	bp	train	7.084	5.573	5.728
"	—	test	7.148	5.428	5.433
•(0)3	bp	train	1.617	1.617	1.617
"	—	test	1.642	1.642	1.642
1(5)4	cp	train	4.622	1.399	1.610
"	—	test	4.491	1.693	1.797
•(0)4	bp	train	1.244	1.244	1.244
"	—	test	1.305	1.305	1.305
2(5)1	cp	train	6.115	4.159	4.387
"	—	test	5.927	3.648	4.158
2(5)2	cp	train	4.754	1.895	2.109
"	—	test	4.622	1.773	2.016
2(5)3	cp	train	6.285	4.560	4.758
"	—	test	6.267	4.730	4.690
2(5)4	cp	train	4.607	1.292	1.568
"	—	test	4.433	1.484	1.715
3(5)1	cp	train	4.609	1.581	1.768
"	—	test	4.376	1.612	1.766
3(5)2	cp	train	4.746	1.934	2.096
"	—	test	4.391	1.754	1.963
3(5)3	cp	train	4.766	1.795	2.002
"	—	test	4.460	1.735	1.853

Table 2

Table 3

Test of the cleanprop procedure for training multilayer feedforward nets on noisy data. As in Table 2, except that the experimental standard deviations σ_r are chosen from a Gaussian distribution with halfwidth $\mathcal{W} = 10$ MeV. Again the same set of σ_r is used in all runs.

Run	Rule	Mode	$\sigma^{[o]}$	σ	$\sigma^{[tr]}$
1(10)1	cp	train	9.094	2.267	2.752
"	—	test	8.644	2.470	2.828
1(10)1	bp	train	11.200	6.159	6.417
"	—	test	10.938	6.274	6.280
•(0)1	bp	train	1.186	1.186	1.186
"	—	test	1.157	1.157	1.157
1(10)2	cp	train	9.916	4.435	4.756
"	—	test	8.994	3.745	4.355
•(0)2	bp	train	3.158	3.158	3.158
"	—	test	2.782	2.782	2.782
1(10)3	cp	train	8.955	1.461	1.903
"	—	test	8.404	1.718	1.874
1(10)3	bp	train	1.708	1.708	1.708
"	—	test	1.869	1.869	1.869
1(10)4	cp	train	9.120	2.315	2.685
"	—	test	8.552	2.206	2.742
•(0)4	bp	train	2.564	2.564	2.564
"	—	test	2.529	2.529	2.529
2(10)1	cp	train	8.532	1.366	1.960
"	—	test	8.513	1.706	2.086
2(10)1	bp	train	11.012	6.188	6.385
"	—	test	10.822	5.752	6.128
2(10)2	cp	train	9.798	4.942	5.081
"	—	test	9.712	4.093	4.427
2(10)3	cp	train	8.432	0.949	1.482
"	—	test	8.429	1.122	1.324
2(10)4	cp	train	8.660	2.048	2.733
"	—	test	8.681	2.671	2.688
3(10)1	cp	train	8.840	1.613	2.159
"	—	test	8.272	1.920	2.307
3(10)1	bp	train	11.389	6.239	6.416
"	—	test	10.731	5.726	6.183
3(10)2	cp	train	10.166	4.845	5.002
"	—	test	9.276	3.962	4.401

Table 3

Table 4

Test of the cleanprop procedure for training multilayer feedforward nets on noisy data. As in Table 2, except that the experimental standard deviations σ_r are chosen from a Gaussian distribution with halfwidth $\mathcal{W} = 15$ or 25 MeV. Again the same set of σ_r is used in all runs for given $\mathcal{W}$.

Run	Rule	Mode	$\sigma^{[o]}$	σ	$\sigma^{[tr]}$
1(15)1	cp	train	13.263	0.717	1.504
"	—	test	12.178	1.021	1.447
1(15)1	bp	train	14.138	5.762	6.117
"	—	test	13.495	5.753	5.936
1(15)2	cp	train	13.460	2.678	3.308
"	—	test	12.578	3.517	3.366
1(15)2	bp	train	14.389	6.177	6.675
"	—	test	13.836	6.496	6.480
1(15)3	cp	train	13.497	2.972	3.603
"	—	test	12.606	2.958	3.799
2(15)1	cp	train	13.620	0.901	1.901
"	—	test	11.939	1.030	2.115
2(15)2	cp	train	13.864	2.381	3.122
"	—	test	11.865	3.322	3.135
2(15)3	cp	train	14.253	3.925	4.224
"	—	test	12.309	4.674	4.507
3(15)1	cp	train	12.957	0.470	1.284
"	—	test	12.198	1.056	1.109
3(15)2	cp	train	13.197	2.530	3.215
"	—	test	12.396	3.198	3.236
3(15)3	cp	train	12.995	0.737	1.475
"	—	test	12.281	1.064	1.650
1(25)1	cp	train	22.051	1.027	1.921
"	—	test	20.241	0.860	1.927
1(25)2	cp	train	22.193	2.232	3.217
"	—	test	20.498	2.463	3.409
1(25)3	cp	train	22.026	1.126	1.867
"	—	test	20.184	2.018	1.875

Table 5

Application of the “cleanprop” algorithm to realistic modeling of experimental mass data. A hybrid binary-analog input coding scheme is used, with each of Z and N coded in binary by eight on-off neurons, and $A = N + Z$ and $N - Z$ coded as floating-point numbers by two dedicated analog neurons [7]. The data base consists of the full set of 1654 current experimental nuclear masses with associated standard deviations $\sigma_e^{(r)} = \sigma_r$. The training set is a randomly selected subset of 1323 nuclides, the remaining 331 examples being used to test prediction. Networks are all of type ${}^h(18 + 10 + 10 + 10 + 1)_a[421]$. Net IIc evolves from IIb, and IIb from IIa, under training with the indicated learning rates η . Connections were not pruned. Quoted results for the conventional rms error measure $\sigma^{[o]}$ of Eq. (10) and the derived theoretical error measure σ of Eq. (13) correspond to the minimum σ found during the 2000 runs at each learning rate. Units of σ and $\sigma^{[o]}$ are MeV.

Net	η	$\sigma(\text{train})$	$\sigma^{[o]}(\text{train})$	$\sigma(\text{test})$	$\sigma^{[o]}(\text{test})$
I	0.5	0.996	1.003	1.483	1.491
IIa	0.5	0.912	0.932	1.174	1.179
IIb	0.25	0.802	0.816	1.092	1.094
IIc	0.1	0.767	0.782	1.060	1.062

Figure Captions

Fig. 1. Training and test nuclides for the problem of fitting the liquid-drop mass formula in the presence of noise.

Fig. 2. Check of the validity of key assumption (I) that the model errors satisfy $\epsilon_m^{(r)} \in N_m(0, \sigma_m^2)$. This example refers to the training set in the case 1(10)4 of table 3.

Fig.1a

Fig.1b

Fig.2