Artificial neural networks for computing in plasma physics

W.L. Morgan†
Kinema Research
18720 Autumn Way
Monument, CO 80132 USA

The concept and computational algorithms of Artificial Neural Networks are described and examples of their application to electron swarm analysis and plasma spectroscopy are presented.

1 INTRODUCTION: MODELING AND SIMULATION IN APPLIED PHYSICS

The use of models to describe physical systems and processes has always been an integral part of physical science. With the invention and widespread use of digital computers and subsequent advances in the speed and processing capacity of computers modeling has become quite complex and sophisticated. Frequently the word simulation is applied to that kind of modeling that attempts to mimic in some direct way the physical behavior of a system as distinct from solving differential initial or boundary value equations that themselves are the model of the system. An example would be the

---

†Visiting Scientist, 1991–92
CNRS (Unité de Recherche Associée 277)
Centre de Physique Atomique
Université Paul Sabatier
31062 Toulouse CEDEX FRANCE

simulation of classical trajectories for electrons moving in a gas in order to compute transport coefficients rather than solving Boltzmann's integro–differential equation for the same process.

Modeling and simulation of the physics of plasmas involves one or more of the standard three levels of description that we find in statistical physics. These levels are (1) the microscopic level, which is described by the equations of motion of the individual particles; (2) the transport level, in which the behavior of ensembles of particles are described statistically by a distribution or density function and a transport equation, such as Boltzmann's equation; and (3) the continuum level, in which the system is described by thermodynamic variables and a set of partial differential equations, such as the Navier–Stokes equations. As I will discuss later, the quantities commonly measured in laboratory plasmas are the transport coefficients and thus, over the years, much effort has been put into working with the transport level of description. When simulations are performed on such plasmas at the microscopic level the procedure is to statistically sample the density function $\rho(\mathbf{r},\mathbf{v},t)$ and use it to compute transport coefficients.

A common feature of modeling and simulation of physical systems and processes using any of these levels of description is that it is usually performed using numerical adaptations of the conventional mathematical descriptions used in physics. That is, the processes are described by ordinary or partial differential equations that are then solved using a numerical finite difference or finite element algorithm. There are other approaches to computing in physics that are in use or are being developed, such as Monte Carlo sampling using the Metropolis algorithm[1] or the path integral molecular dynamics of Car and Parrinello;[2] cellular automata,[3] which are rule based Boolean extensions of the lattice gas methods of statistical mechanics; genetic algorithms;[4] and artificial neural networks[5-7], the subject of this paper.

2 NEURAL NETWORKS

2.1 Learning and prediction

Jake (a 5 year old): "Dad, why when you throw things in the basket don't they miss?"

Me: "Do you miss?"

Jake: "Yeah, lots of times."

When most of us throw an object toward a target we do not solve the equations of

motion **F** = M**a** as part of the process, yet we have predicted the trajectory and have applied the force appropriate to Newton's second law in order to land the object at or near the target. By some age, apparently greater than five years, we have learned ballistics even if we are not able to write down and solve the mathematical equations describing ballistics. We have learned enough from observation of a finite set of experiments to attempt to predict the result of a new, but not too different, experiment. We could say that given a set of observations of the output vector **y** for a given input vector **x** in a finite series of n experiments:

$$(\mathbf{x}_1,\mathbf{y}_1), (\mathbf{x}_2,\mathbf{y}_2), \ldots , (\mathbf{x}_n,\mathbf{y}_n)$$

we have learned enough about the unknown functional relationship $\mathbf{y} = f(\mathbf{x})$ to predict **y** when presented with a new **x** not belonging to the known set. This kind of learning in order to develop a predictive capability is what our brains do and what artificial neural networks are designed to do.

2.2 Artificial neural networks

Based on notions of how human brains are structured artificial neural networks (ANN) consist of layers of simulated "neurons" with associated activation functions, transfer functions, and weighting functions for the "synapse" connections to other neurons. A typical configuration for an ANN is shown in Fig. 1. The key elements are an input layer, one or more "hidden" layers, and an output layer. Each neuron has a transfer function associated with it, typically the sigmoid function $T(x)=1/(1+e^{-x})$, that gives an output value that is a non–linear function of the sum of the input values. The input values are the weighted outputs of each neuron in the previous layer. That is, if the output of neuron j is o_j and w_{ij} is the weight connecting neurons i and j, then the output of neuron i is:

$$o_i = T(\sum_j w_{ij} o_j) = 1/(1 + e^{-\sum_j w_{ij} o_j})$$

where the sum is over all neurons j having outputs that feed into neuron i. The hidden layers give the network a high degree of non–linearity and, from the point of view of the network as a pattern matcher, provide an internal representation of the correlation between the input and the output patterns. The concept behind this kind of network (known as a *feed–forward, back–propagation* network) is that it can "learn" to associate a set of output

patterns with a set of input patterns by adjusting the weights that connect together the network of non–linear devices. One might also think of a neural network as a highly flexible many parameter interpolation and, hence, extrapolation algorithm.

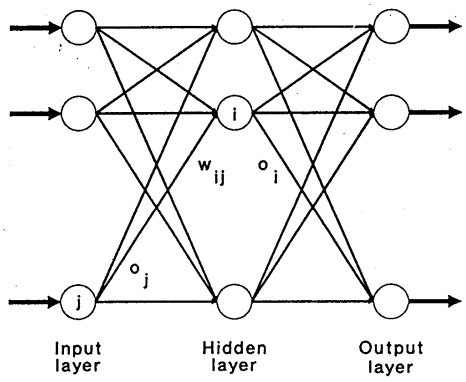

Figure 1: Three–layer feedforward, backpropagation neural network. The neurons, denoted by circles, in a given layer operate via a transfer function on the weighted output from the neurons in the previous layer to produce an output signal that is passed on to the next layer.

A neural network might, for example, be constructed for learning and then predicting solutions to Laplace's equation. If we are given the temperature $T_0(x,y)$ along the boundary of a two–dimensional area the temperature within the boundary is given by the solution to $\nabla^2 T = 0$. One could conceivably perform a number of measurements of T at points along the boundary and at some interior points and train a neural network on this information where the input neurons would represent the temperatures at a finite set of points on the boundary and the the output neurons would be the interior temperatures. Then when presented with a new distribution of boundary temperatures the ANN should be able to predict the interior temperatures. Observe that this process of prediction is model independent in that we have assumed no knowledge of Laplace's equation and its solutions. We could also compute a list of solutions of Laplace's equation for a number of different boundary conditions, what we call synthetic data; train a neural network on these synthetic data; and use it to predict the interior temperatures when presented with the

boundary temperatures. This, of course, is model dependent and the results can be no better than the model used to produce the synthetic data.

Finally, returning to the ballistics problem used to introduce the notion of learning, researchers at MIT have constructed a robot arm holding a flat plate on which a ball can be bounced and have developed a neural network controller for it that has been trained to keep the ball bouncing and confined to the area of the plate!

2.3 Training neural networks

Neural networks have a "learning" capability in that the weights associated with connections between pairs of neurons can be modified (strengthened or weakened) in response to the network's successes and failures so as to optimize in favor of the network's successful strategies. The network is trained by running a number of cases of known {input,output} sets through it and adjusting the weights to minimize the sum of the squares of the differences between the desired result and the computed result. This quadratic function is the so–called *energy*, *cost*, or *objective* function. The weights are adjusted using what is known as the generalized delta rule.[5-7] The energy function, E, can be written

$$E = \sum_p \sum_i [t_i(p) - o_i(p)]^2 ,$$

where $t_i(p)$ is the value of the training output for output neuron i and training data set p; $o_i(p)$ is the corresponding network output. E is minimized using an iterative procedure whereby the weights w_{ij} are adjusted according to:

$$\Delta w_{ij} = \sum_p \epsilon \delta_i(p) o_j(p) ,$$

where $\delta_i(p) = [t_i(p) - o_i(p)] o_i(p) [1 - o_i(p)]$ for output layer neurons,

and $\delta_i(p) = o_i(p)[1 - o_i(p)] \sum_j w_{ij} \delta_j(p)$ for hidden layer neurons.

The parameter ϵ is called the training rate coefficient. The weights are adjusted starting with the neurons in the output layer and moving back a layer at a time toward the input layer.

The ANN will assimilate and correlate data given it in the process of "training" so that when given input vectors outside of the training set it will produce reasonable values

for the corresponding output vectors. The matrix of weights, w_{ij}, clearly represents the mapping between the set of input vectors and the set of output vectors and contains all the information correlating the output to the input.

3 NEURAL NETWORK APPLICATIONS

To date there have been few publications in the applied physics literature involving neural networks. The best understood and most readily applicable networks are the backpropagation networks using supervised training that I have been discussing here. They are commonly used for pattern recognition problems[8] [9], such as optical character recognition or signal analysis, and may have similar applications in particle track identification in high energy physics. Other published demonstrations of the potential use of neural networks include classification of complex atomic energy levels,[10] noisy signal identification in radiology,[11] predicting solutions to Schrödinger's equation,[12] time series prediction,[13] and inversion of remote sensing data.[14]

3.1 A neural network for obtaining collision cross sections from transport data

3.1.1 Cross sections, transport coefficients, and Boltzmann's equation

The use of theory to obtain collision cross sections from electron transport data was pioneered by Townsend and by Ramsauer in the 1920's. The method used in such early analyses involved measuring the drift velocity of electrons in a gas as a function of E/p (electric field strength divided by gas pressure) and inverting the integral relating the drift velocity and the momentum transfer cross section using an approximate expression for the energy distribution of the electrons. This technique has increased in sophistication over the years. In the 1960's Phelps and various collaborators applied electronic computation to the problem and developed algorithms for solving Boltzmann's equation for transport of electrons in a weakly ionized plasma to obtain an accurate electron energy distribution function valid at higher fields and in the presence of inelastic and, even, superelastic collisions. This began an era that has given us very accurate momentum transfer and lower energy (rotational and vibrational) inelastic cross sections that have been derived from measurements of the drift and diffusion of electron swarms in gases. This methodology is reviewed in Refs. [15–18]. This has become an increasingly active field in recent years due, for example, to the desire for cross sectional data on molecules such as CH_4, CF_4, SF_6,

SiH_4, and SiF_4 that are used in semiconductor plasma processing and in switching applications.

The cross sections are fundamental quantities depending only on the energy of the incident electron and the particular initial and final atomic or molecular states. The swarm parameters, such as drift velocity, diffusion coefficients, and excitation coefficients, are derived quantities depending on the local environment of the electron swarm. They are integrals over the product of a cross section and the energy distribution $f(\epsilon)$ of the electrons, which is the solution to Boltzmann's transport equation.

The process of inverting the swarm data to obtain cross sections has involved inserting cross section models in the collision terms of Boltzmann's equation, calculating $f(\epsilon)$ and the swarm coefficients, altering the model cross sections, and iterating until an acceptable match between measured and computed swarm coefficients is found. Clearly this iterative process is very labor intensive. The experience of the researcher plays an important role comparable to that of the specific computational techniques used. I have investigated[19] [20] the use of numerical optimization algorithms as a means of aiding in obtaining cross sections from electron swarm data. Here I describe the use of a *neural network* to explore the mapping between swarm coefficients and cross sections.[19]

The relationship between the cross sections and the transport coefficients via the distribution function $f_0(\epsilon)$ is highly nonlinear. We have a mapping

$$\left\{ \begin{array}{c} \sigma_m(\epsilon) \\ \\ \{\sigma_i(\epsilon)\} \end{array} \right\} \longrightarrow \left\{ \begin{array}{c} v_d(E/N) \\ D/\mu(E/N) \\ \{k_i(E/N)\} \end{array} \right\}$$

and we want to find the reverse mapping given the transport data Here σ_m is the momentum transfer cross section; the σ_i are the various inelastic cross sections; v_d is the electron drift velocity; D/μ is the characteristic energy; and the k_i are rate coefficients for the inelastic processes.

The mapping function is Boltzmann's equation for electron transport in a gas:

$$(\partial/\partial t + \mathbf{v}\cdot\nabla_r + \frac{e\mathbf{E}}{m}\cdot\nabla_v)f(\mathbf{r},\mathbf{v},t) = (\partial f/\partial t)_{\text{collisions}} \ ,$$

If we neglect the spatial and temporal dependence of the distribution function $f(\mathbf{r},\mathbf{v},t)$, and

express f=f(**v**) as the first two terms of a spherical harmonic expansion, that is

$$f(\mathbf{v}) = f_0(v) + \frac{\mathbf{v}}{v}\cdot\mathbf{f}_1(v) \ ,$$

then we obtain [see Ref. 16 for details of the derivation] the following scalar equation for $f_0(\epsilon)$ (where $\epsilon=mv^2/2$):

$$\tfrac{1}{3}(eE/N)^2\, d/d\epsilon\{\epsilon/\sigma_m\, df_0/d\epsilon\} + d/d\epsilon\{(2m\sigma_m/M)\epsilon^2[f_0(\epsilon) + kT\, df_0/d\epsilon]\}$$
$$+\sum_i [(\epsilon+\epsilon_i)\sigma_i(\epsilon+\epsilon_i)f_0(\epsilon+\epsilon_i) - \epsilon\sigma_i(\epsilon)f_0(\epsilon)] = 0.$$

Here we have assumed that the populations of the excited levels, labeled by i, are small enough that superelastic collisions and transitions among excited states are unimportant. This is the form of Boltzmann's equation that is most often solved in the iterative process of developing a set of cross sections given a collection of transport coefficients. The two most commonly measured transport coefficients are the drift velocity, v_d, and the transverse diffusion coefficient, D, which are related to $f_0(\epsilon)$ and the momentum transfer cross section, $\sigma_m(\epsilon)$ by the following:

$$v_d \propto \int[\sigma_m(\epsilon)]^{-1}(df_0/d\epsilon)\epsilon d\epsilon \quad \text{and} \quad D \propto \int[\sigma_m(\epsilon)]^{-1}f_0(\epsilon)\epsilon d\epsilon.$$

The drift velocity and diffusion coefficient sample different aspects of $f(\epsilon)$ and, hence, represent two somewhat independent pieces of information. If they are available we might also make use of measured rate coefficients, $k_i \propto \int\sigma_i(\epsilon)f_0(\epsilon)\epsilon d\epsilon$, and spectral data.

3.1.2 Xe momentum transfer cross section

Initially I applied the neural network technique to a model problem and then extended it to estimate the momentum transfer cross section for xenon from the measured drift velocities and characteristic energies of electrons in xenon. In this application the input neurons represent swarm coefficients as functions of E/N and the output neurons represent the momentum transfer cross section as a function of electron energy. The neural network is schematically diagramed in Fig. 2. The problem of inverting the swarm data has been transformed into a pattern matching problem where the pattern is the mapping

between $\{v_d(E/N), D/\mu(E/N)\}$ and $\{\sigma_m(\epsilon)\}$.

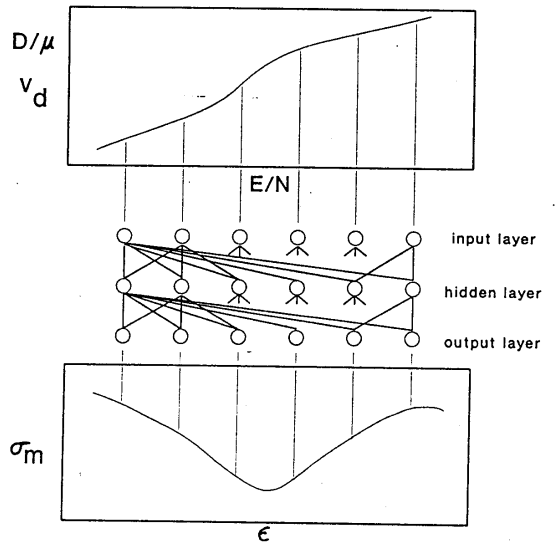

Figure 2: Schematic diagram of the relationship between the swarm data input, the neural network, and its cross section output.

I trained a conventional feedforward, backpropagation neural network on 25 sets of $\{v_d(E/N), D/\mu(E/N)\}$ data for cross sections of the form $\sigma(\epsilon) = \sigma_0\epsilon^p$ where $-1 \leq p \leq +1$. That is, we have some cross sections that increase with energy and some that decrease with energy. I then constructed an input set for Xe with $\{v_d(E/N)\}$ from Hunter, et al.[21] and $\{D/\mu(E/N)\}$ from Koizumi, et al.[22] Unfortunately neither paper presented *both* drift velocity *and* characteristic energy data. I found that I was able to obtain reasonable results with a network of only three layers:

Layer	Neurons	Weights
1	18	–
2	20	380
3	9	189

How many neurons and layers one needs for a given problem is discussed in the concluding section.

The cross section that the neural network returned in the output layer for xenon in the energy range around the Ramsauer minimum is shown in Fig. 3 along with the $\sigma_m(\epsilon)$ from Hunter, et al.,[21] Koizumi, et al.,[22] and Frost and Phelps.[23] These authors' results were derived from either drift velocity or characteristic energy measurements. We see that the neural network gives a respectable estimate of the cross section even though the number of E/N values is small and the energy grid is very coarse. This is a stringent test because of the Ramsauer minimum; the cross section is not monotonically decreasing or increasing as are the training data. This illustrates the generalizing ability of the neural network.

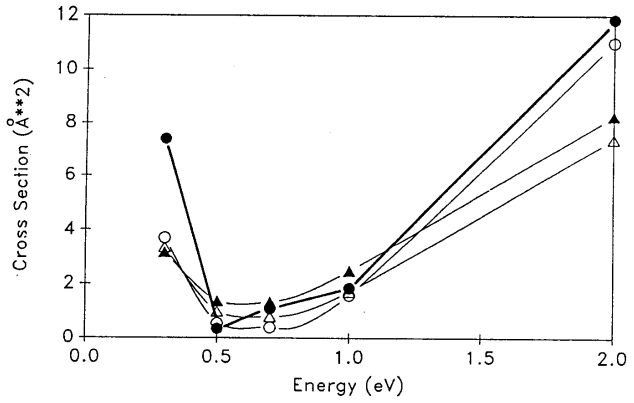

Figure 3: Xe momentum transfer cross section. The filled circles are the output from the neural network; the open circles are from Hunter, et al.;[21] the open circles are from Koizumi, et al.;[22] and the filled triangles are from Frost and Phelps.[23]

There is little in the way of previous experience in applying neural networks to the problems of physics to serve as guidance in constructing neural networks for problem such as this. One finds in the literature rules of thumb such as rarely needing more than two

hidden layers or using for the number of hidden layer neurons the average of the number of input layer and output layer neurons. There are numerous training algorithms that take varying amounts of processor time and memory but the only way of determining which ones work best is to try them. These calculations were performed on an 80386/87 personal computer. Typical training times were between a half hour and an hour, which, depending on the number of neurons in the network, corresponds to hundreds to several thousand cycles through the set of training data. Of course, for a given input the computation of the network output takes only a fraction of a second.

3.2 Neural networks for plasma spectroscopic analysis

The standard technique for finding the temperature and density of high temperature plasmas, such as those found in stars and in laboratory magnetically confined or inertially confined fusion plasmas, has long been spectroscopy.[24 25] One measures the intensities and widths of a number of spectral lines being emitted by the plasma and compares the results with model calculations in order to deduce T_e and N_e. The technique is obviously strongly model dependent but there is no alternative. We[26] have been developing a neural network for use in plasma x–ray spectroscopic analysis of high density laser produced plasmas.

In developing the neural network techniques for this application we are considering the K–shell spectra of a laser produced aluminum plasma. These spectra consist of a number of lines that are emitted by hydrogen–like and helium–like aluminum ions in the plasma. Measurements of spectra of such plasmas can be found in Refs. 27 and 28. The spectral lines that we are interested in and for which we have measurements are the H_α, H_β, H_γ, H_δ, He_α, He_β, He_γ, He_δ, and He_{ic}. This last is the so–called helium intercombination line, which is the $1s2p\ (^3P_1) \rightarrow 1s^2\ (^1S_0)$ transition in He–like Al. We use line ratios, such as $I(H_\alpha)/I(He_\beta)$, rather than absolute intensities as input to the neural network; the network consequently has eight input neurons. The two output neurons are T_e and N_e. The neural network that we are currently working with has two hidden layers each having 15 neurons.

We are presently using a plasma modeling program called RATION[29] to generate the synthetic data used in training the neural network. Using RATION we compute 95 training patterns spanning the space of $50 \leq T_e \leq 1200$ eV and $5\text{x}10^{19} \leq N_e \leq 7\text{x}10^{21}$ cm^{-3}. The network has been trained on the synthetic data to a tolerance of 10%. An example of the kind of results that we then obtain by presenting the trained network with synthetic "mystery" line intensity data generated using RATION is shown in the following table

with T_e in eV and N_e in units of 10^{20} cm^{-3}:

Network		Correct	
T_e	N_e	T_e	N_e
218	2.3	230	2.5
236	2.7	250	3.0
299	2.8	310	3.0
369	4.7	370	5.0
437	5.8	430	6.0

The results obtained using the measured aluminum x–ray spectra are not as good, but this is where the model dependence becomes a critical factor. In this context we are clearly using the neural network as a multi–dimensional interpolation algorithm. Problem areas where it is hoped that the neural network technique will be advantageous involve noisy data and partial data. In addition there is the issue of how far out of local thermodynamic equilibrium the plasma is and whether the network has been trained on the appropriate models. The right choice of input neurons and training data that reflect the range of non–LTE possibilities should allow the network to handle the very likely event of different sets of line ratios having the same T_e and N_e. The generalizing capability of neural networks should manifest itself when dealing with these issues.

A further advantage of using neural networks on a problem such as this one is that once the network is trained, however long that may take (this one, which is written in the Mathematica language,[30] took about 5 hours to train on a NeXT workstation), it can compute an output for a new input in milliseconds. In addition, all the information resides in the matrix of weights so that, even if it takes many hours or days to train the network on a supercomputer, the trained network can be run on any computer.

4 CONCLUSION

The neural network configuration discussed above is known as a feedforward, backpropagation network. It is feedforward because all the operations consist of layer n+1 operating on the output of layer n and backpropagation because the training algorithm adjusts the weights from the output layer working back toward the input layer. We have seen that such networks, when properly trained, are capable of many dimensional

interpolation and, beyond this, generalization. They are capable of dealing with noisy or even missing data. Clearly the basic numerical algorithm for the training and operation of the network is quite simple. The difficult part of applying neural network techniques to a problem lies in casting the problem in an appropriate form, choosing the right variables as input and as output, and making the right choice of training data.

The conventional wisdom has been that neural networks are useful for only very rough solutions and not for accurate scientific calculations but some authors, such as Lapedes and Farber[13] refute that point of view. Indeed, the claim has been made[31] that neural networks are "formally capable of solving any conventional computational problem." There are limitations, however, due to computer hardware. Our brains have of the order of 10^{12} neurons and 10^{13} connections whereas we are limited to thousands of neurons and tens of thousands connections on general purpose digital computers. There are some gains that can be achieved using specially designed neural computers[6] and, perhaps in the future, optical neural computers.[9]

I have not mentioned in this discussion the other kinds of neural networks and training algorithms in existence. It is possible, for example, for the network to fall into a local minimum, which can be avoided by use of a statistical training algorithm using the computational technique of simulated annealing,[32] [33] a variation on the Metropolis algorithm. Such a network is called a Boltzmann machine.[34] [35] There is a recursive version of the feedforward, backpropagation network in which the values from the output layer are fed back to the input layer. There is also a class of free running recursive networks, called Hopfield networks,[36] that do not involve training. Such a network has been used[37] to find good solutions to the celebrated traveling salesman problem, a complicated minimization problem.[33] As this area of research is very much in its infancy, we can expect many new developments in the understanding of neural networks and in their applications to the problems of physics in the future.

5 ACKNOWLEDGEMENTS

My collaborator on the application of neural networks to plasma spectroscopy is Dr. Jon T. Larsen of Cascade Applied Sciences in Boulder, Colorado. I would like to thank Savino Longo, Dr. Barbara L. Whitten, and Dr. Bill Goldstein for helpful discussions about neural networks as well as Jacob L.W. Morgan for his observations on learning Newton's second law. Many of the calculations have been performed on a NeXT workstation at The Colorado College in Colorado Springs. The research has been partially funded by the Lawrence Livermore National Laboratory, Livermore, California.

6 REFERENCES

1. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys. **6**, 1087 (1953); see also the review by G. Bhanot, Rep. Prog. Phys. **51**, 429 (1988).
2. R. Car and M. Parrinello, Phys. Rev. Letts. **60**, 204 (1988).
3. T. Toffoli and N. Margolus, **Cellular Automata Machines**, (MIT Press, Cambridge, MA, 1987).
4. D.E. Goldberg, **Genetic Algorithms in Search, Optimization, and Machine Learning**, (Addison–Wesley, Reading, MA, 1989).
5. P. D. Wasserman, **Neural Computing**, (Von Nostrand Reinhold, New York, 1989).
6. R. Hecht–Nielsen, **Neurocomputing**, (Addison–Wesley, Reading, MA, 1990).
7. J. Hertz, A. Krogh, and R.G. Palmer, **Introduction to the Theory of Neural Computation**, (Prentice Hall, Englewood Cliffs, NJ, 1992).
8. B. Kosko, **Neural Networks and Fuzzy Systems**, (Prentice Hall, Englewood Cliffs, NJ, 1992).
9. B. Kosko, **Neural Networks for Signal Processing**, (Prentice Hall, Englewood Cliffs, NJ, 1992).
10. K.L. Peterson, Phys. Rev. A **41**, 2457 (1990).
11. J.M. Boone, V.G. Sigillito, and G.S. Shaber, Med. Phys. **17**, 234 (1990).
12. J.A. Darsey, D.W. Noid, and B.R. Upadhyaya, Chem. Phys. Letts. **177**, 189 (1991).
13. A. Lapedes and R. Farber, "How neural nets work," in **Neural Information Processing Systems**, ed. D.Z. Anderson, (Amer. Inst. of Physics, New York, 1988), pp. 442–456.
14. W. Jeffrey and R. Rosner, Astrophys. J. **310**, 473 (1986).
15. A.V. Phelps, Rev. Mod. Phys. **40**, 399 (1968).
16. I.P. Shkarofsky, T.W. Johnston, and M.P. Bachynski, **The Particle Kinetics of Plasmas**, (Addison–Wesley, Reading, MA, 1966).
17. L.G.H. Huxley and R.W. Crompton, **The Diffusion and Drift of Electrons in Gases**, (Wiley, New York, 1974).
18. A.V. Phelps "Relations between electron–molecule scattering and swarm experiments and analysis," in **Swarm Studies and Inelastic Electron Molecule Collisions**, ed. L.C. Pitchford, B.V. McKoy, A. Chutjian, and S. Trajmar, (Springer–Verlag, New York, 1987), pp. 127–141.
19. W.L. Morgan, IEEE Trans. on Plas. Sci. **19**, 250 (1991).
20. W.L. Morgan, Phys. Rev. A **44**, 1677 (1991).

21. S.R. Hunter, J.G. Carter, and L.G. Christophorou, Phys. Rev. A **38**, 5539 (1988).
22. T. Koizumi, E. Shirakawa, and I. Ogawa, J. Phys. B **19**, 2331 (1986).
23. L.S. Frost and A.V. Phelps, Phys. Rev. **136**, A1538 (1964).
24. H. Griem, **Plasma Spectroscopy**, (McGraw–Hill, New York, 1964).
25. D. Mihalas, **Stellar Atmospheres**, (W.H. Freeman, San Francisco, 1978).
26. J.T. Larsen, W.L. Morgan, and W.H. Goldstein, "Artificial neural networks for plasma x–ray spectroscopic analysis", Rev. Sci. Instr., to be published.
27. B.K.F. Young, et al., Phys. Rev. Letts. **61**, 2851 (1988).
28. B.K.F. Young, et al., J. Phys. B **22**, L533 (1989).
29. R.W. Lee, B.L. Whitten, and R.E. Strout II, J. Quant. Spec. and Rad. Transfer **32**, 91 (1984).
30. S. Wolfram, **Mathematica**, (Addison–Wesley, Redwood City, CA, 1991).
31. Y.S. Abu–Mostafa, "Neural networks for computing?," in **Neural Networks for Computing**, ed. J.S. Denker, (American Institute of Physics, New York, 1986), pp. 1–6.
32. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, Science **220**, 671 (1983).
33. W.H. Press,B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, **Numerical Recipes**, (Cambridge Univ. Press, Cambridge, 1986).
34. G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in **Parallel Distributed Processing**, vol. 1, ed. D.E. Rumelhart and J. L. McClelland, (MIT Press, Cambridge, MA, 1986), pp. 282–317.
35. E. Aarts and J. Korst, **Simulated Annealing and Boltzmann Machines**, (Wiley, New York, 1989).
36. J.J. Hopfield, Proc. Nat. Acad. Sci. **79**, 2254 (1982); **81**, 3088 (1984).
37. J.J. Hopfield and D.W. Tank, Biological Cybernetics **52**, 141 (1985).