Recurrent Neural Networks

Networks for Temporal Pattern Learning

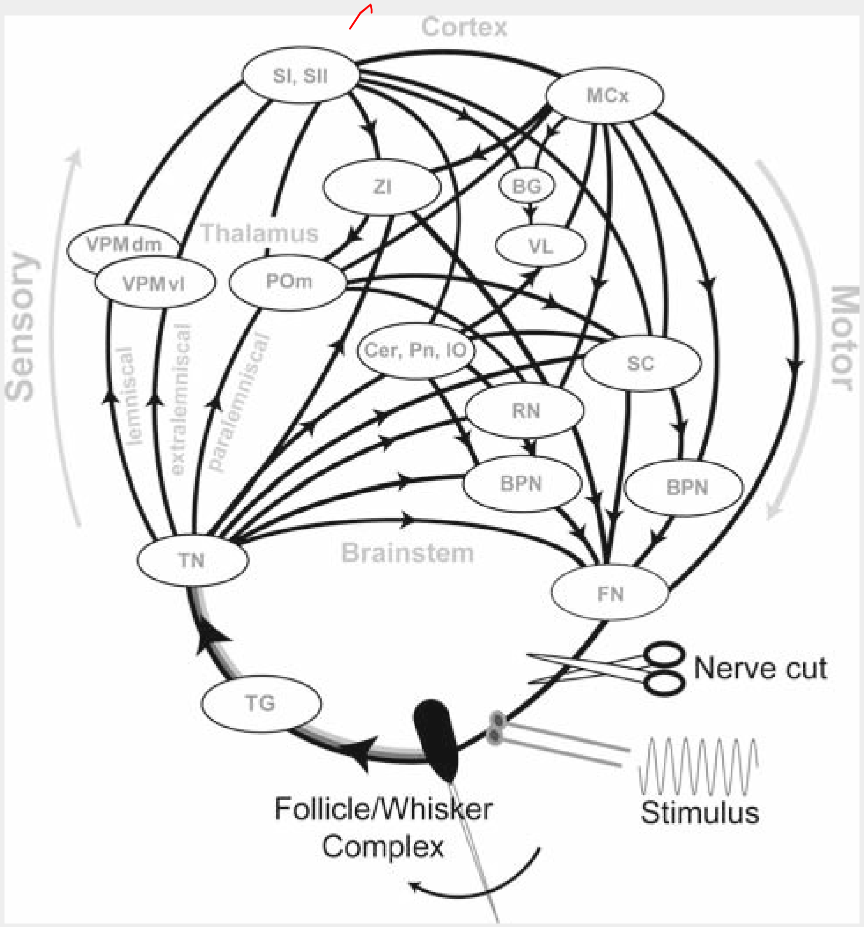
Dr J. Charles Sullivan

Networks with Internal State

- All of the neural networks we have looked at so far are ‘stimulus response’
  - the output is only dependent on the current inputs
- However, in many cases, we need *internal state*
  - the output to be dependent on previous states of the system
- The word ‘system’ covers the process of generating the inputs to the neural network (the ‘plant’) or the final outputs of the neural network, or internal neuron outputs within the neural network

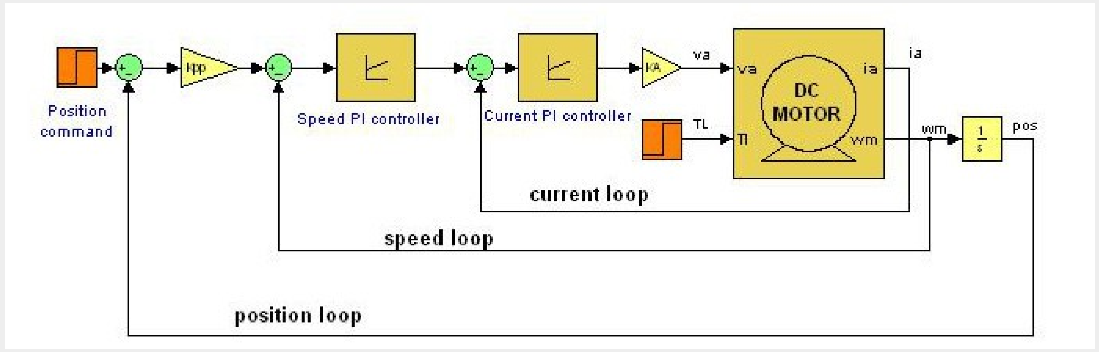
Recurrent Connections

- All of the networks seen so far have been feed-forward
  - Connections all directed from inputs to outputs ie looped
- Biological networks aren't like that
  - Lots of feedback loops
  - Nested loops
  - On the right, for example, is a model of the rat whisker sensory-motor system



A Hypothetical Control Example

- say that we want a neural network to carry out angular position control of an electric motor, as in the PID controller shown here



- An MLP ANN with a single output will provide the voltage to the motor.
- The feedback signals from the motor are current, angular velocity and angular position
- if we want to make a neural network response dependent on a fixed number of past outputs of the plant, then we could store a fixed sized 'window' set of these past outputs and present them as if they are part of a wider set of current plant outputs

Control Example (continued)

- we want the ANN to be able to learn a **temporal model** of behaviour in order to improve its performance
  - we decide that, at the sampling time of the computer running the ANN, we need to use the feedback signals from the previous 3 time steps
- this means that an ANN with 12 inputs would be required: 4 current inputs progressively further into the past and, likewise, 4 position and velocity inputs
- we might need to include more past states in the model, and hidden neurons as well, possibly fully connected
- each of these connections would have an associated weight to be learned
- in the ANN structure, we have traded 'time for space'.
  - however, sometimes the number of time steps into the past is too big, is not known, or is variable.
- we could use a neural network that stores previous input states, possibly in a partially processed form, with internally fed-back signals within the neural network structure
  - End up with a fully Recurrent Neural Network (RNN)
  - Training could be a problem (e.g. BPTT)

$\leftarrow$ Back propagation through time.

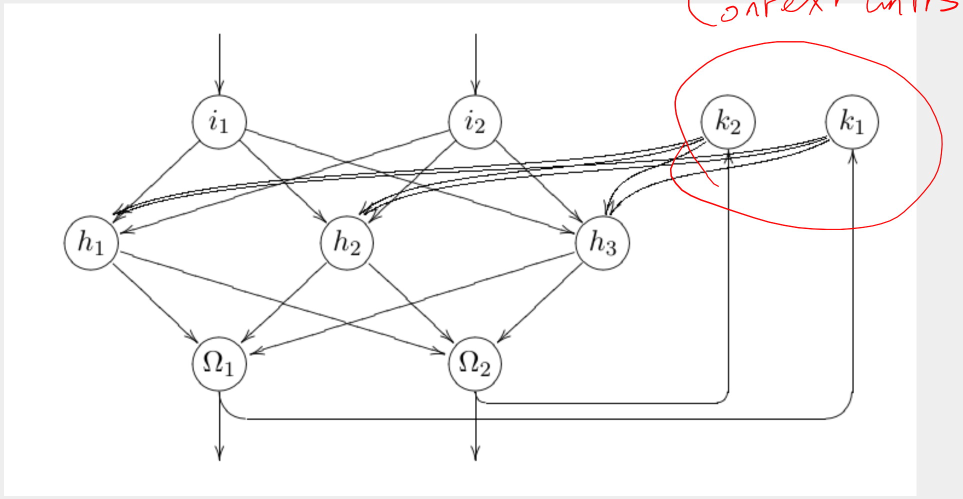
Can we use a simpler model?

- Simple Recurrent Networks
  - Use “context units” which memorize the previous state
  - Limited connections
  - Limited to just one past state
  - Reduced number of adaptive weights
- Two examples
  - Jordan networks
  - Elman networks

Jordan networks

- a multi-layer perceptron with a set K of so-called context neurons k1, k2, . . . , k|K| (Jordan,1986)
- one context neuron per output neuron.
- context neuron just memorises an output until it can be processed in the next time step.
- weighted connection between each output neuron and one context neuron.
- context neuron k_i receives the output value of neuron i at time t and then re-enters it into the network at time (t + 1)

Context units



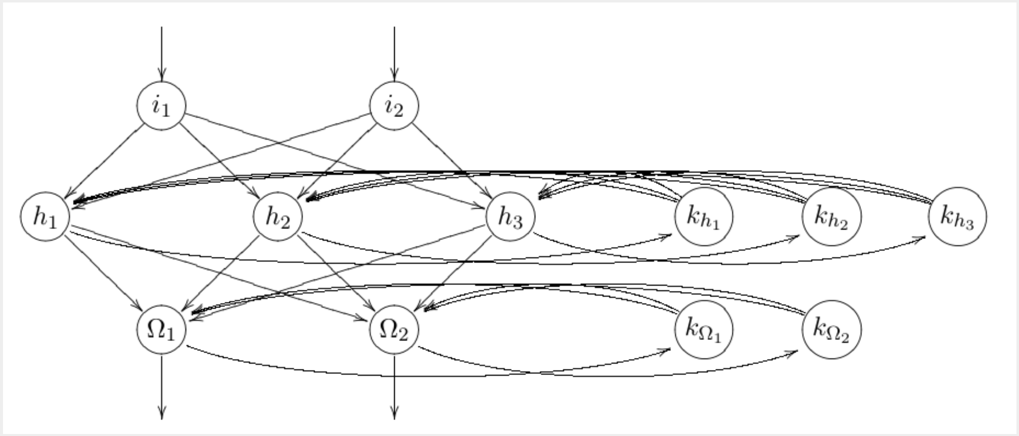
Think of an arm
↳ lots of DOF
lots of muscles
Need for temporal control.

Elman Networks

- Elman networks (Elman,1990) have context neurons, too, but one layer of context neurons per information processing neuron layer

- outputs of each hidden neuron or output neuron are fed into the associated context layer (again exactly one context neuron per neuron)
  - reentered into the complete neuron layer during the next time step
- So the complete information processing part of the MLP exists a second time as a "context version"
  - which considerably increases dynamics and state variety

- Compared with Jordan networks, Elman networks often have the advantage of acting more purposefully since every layer can access its own context



Temporal Pattern Learning

- an Elman (or a Jordan) network given identical inputs at different time steps can give different outputs at these successive time steps due to different feedback states.

- Because the network can store information for future reference, it is able to learn ***temporal patterns*** as well as *spatial patterns*.

USEFUL

Elman network using MATLAB

`elmannet(layerdelays,hiddenSizes,trainFcn)`

takes these arguments:

`layerdelays`	Row vector of increasing 0 or positive delays (default = 1:2)
`hiddenSizes`	Row vector of one or more hidden layer sizes (default = 10)
`trainFcn`	Training function (default = 'trainlm')

and returns an Elman neural network

Using Elman Network in a Time Series Example

Here an Elman neural network is used to solve a simple time series problem

```
[X,T] = simpleseries_dataset;
net = elmannet(1:2,10);
[Xs,Xi,Ai,Ts] = preparets(net,X,T);
net = train(net,Xs,Ts,Xi,Ai);
view(net)
Y = net(Xs,Xi,Ai);
perf = perform(net,Ts,Y)
```

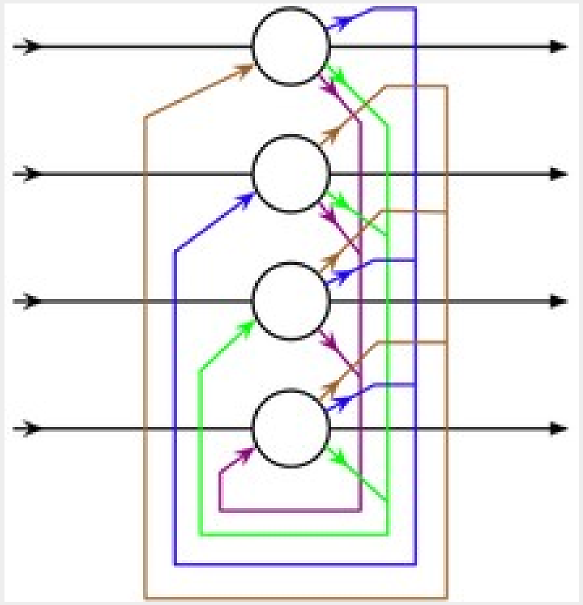
But MATLAB advice is to use a NARX (Nonlinear AutoRegressive network with eXogenous inputs) network instead

Applications of Recurrent Neural Networks (RNN)

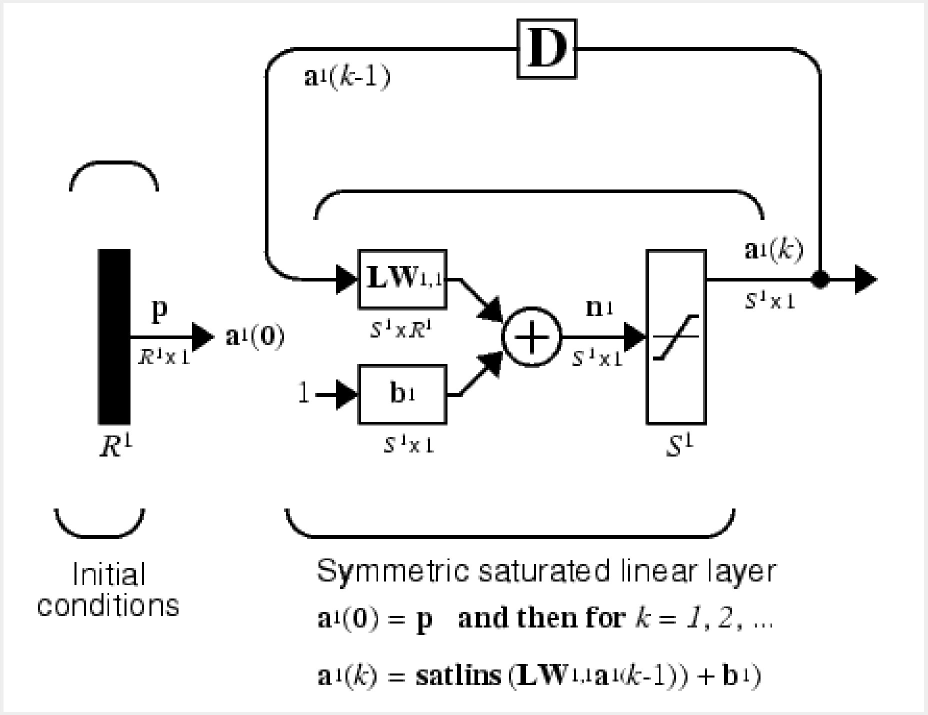
- Originally proposed for time-series modelling and prediction
- e.g. speech recognition (Elman 1990)
- Also applied in signal processing and in a wide variety of robotics control and planning applications (Ziemke 2000)
- Of interest to Cognitive Scientists as models of memory formation

Hopfield Networks

- In 1982 physicist John Hopfield proposed an asynchronous neural network model which made an immediate impact in the AI community (Hopfield, 1982)
- The idea for the Hopfield networks originated from the behavior of particles in a magnetic field: Every particle "communicates" (by means of magnetic forces) with every other particle (completely linked) with each particle trying to reach an energetically favorable state (i.e. a minimum of the energy function).
- It is a special case of a ***bidirectional associative memory.***
- This type of network stores a specific set of equilibrium points such that, when an initial condition is provided, the network eventually comes to rest at such a design point.
- It can, therefore, be used as a form of classifier that groups input vectors into classes. The network is recursive in that the output is fed back as the input, once the network is in operation.
- As the network runs through successive time steps, hopefully, the network output will settle on one of the original design points, i.e., the output will converge on the nearest class that was presented during training.



Hopfield Network Architecture



$a^1(0) = p$ and then for $k = 1, 2, \ldots$

$a^1(k) = \text{satlins}(LW^{1,1} a^1(k-1)) + b^1)$

The structure is similar to the Elman network, in fact even simpler.

However, now the input p to this network merely supplies the initial conditions at time 0, and does not play any further part after this.

This network can be tested with one or more input vectors which are presented as initial conditions to the network.

After the initial conditions are given, the network produces an output which is then fed back to become the only input from then on.

Connectivity of Hopfield Nets

- it is assumed that the individual units preserve their individual states until they are selected for a new update.
  - The selection is made randomly.

- A Hopfield network consists of n fully connected units, that is, each unit is connected to all other units except itself
  - The network is symmetric because the weight w_{ij} for the connection between unit i and unit j is equal to the weight w_{ji} of the connection from unit j to unit i.
  - This can be interpreted as meaning that there is a single bidirectional connection between both units.

- The absence of a connection from each unit to itself avoids a permanent feedback of its own state value.

Energy Function

Hopfield nets are examples of a wider class of dynamical physical systems that may be thought of as instantiating "memories" as stable states associated with minima of a suitably defined system energy.

The energy function of a state $\vec{x} = (x_1, x_2, \ldots, x_n)$

is given by $E(\vec{x}) = -\frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \theta_i x_i$

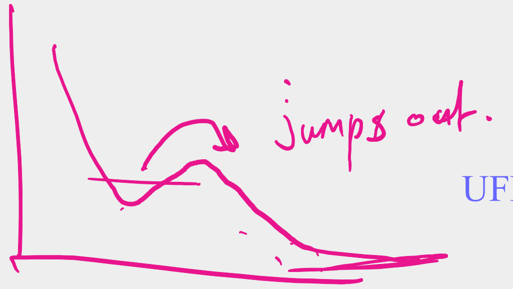
Solving the TSP

Travelling Salesman Problem

- Hopfield nets guarantee that a ***local minimum of the energy function*** will be reached.
- If an optimization problem can be written in an analytical form isomorphic to the Hopfield energy function, it can be solved by a Hopfield network
- Hopfield and Tank (1985) showed that the Hopfield network could be used to solve the Travelling Salesman Problem (TSP)
- In general the network will not be capable of finding the global minimum and because large TSPs (with more than 100 cities) have so many local minima, it is difficult to decide whether the local minimum that has been found is a good approximation to the optimal result
- The whole approach is dependent on the existence of real, massively parallel systems, since the number of units required to solve a TSP increases quadratically with the number n of cities (and the number of weights increases proportionally to n^4)

Problem of Local Minima

- The main difficulty is that complex combinatorial problems produce an ***exponential number of local minima of the energy function***.
- In sequential computers, Hopfield models cannot compete with conventional methods.
  - The only way to make Hopfield models competitive is through the use of special hardware.
- One strategy to avoid local minima of the energy function consists in introducing **noise** into the network dynamics.
  - As usual, the network will reach states of lower energy but will also occasionally jump to states higher up in the energy ladder.
  - In a statistical sense we expect that this extended dynamics can help the network to skip out of local minima of the energy function.
- The best-known models of such stochastic dynamics are **Boltzmann machines**



UFME7K-15-M

Error Correction/Classification

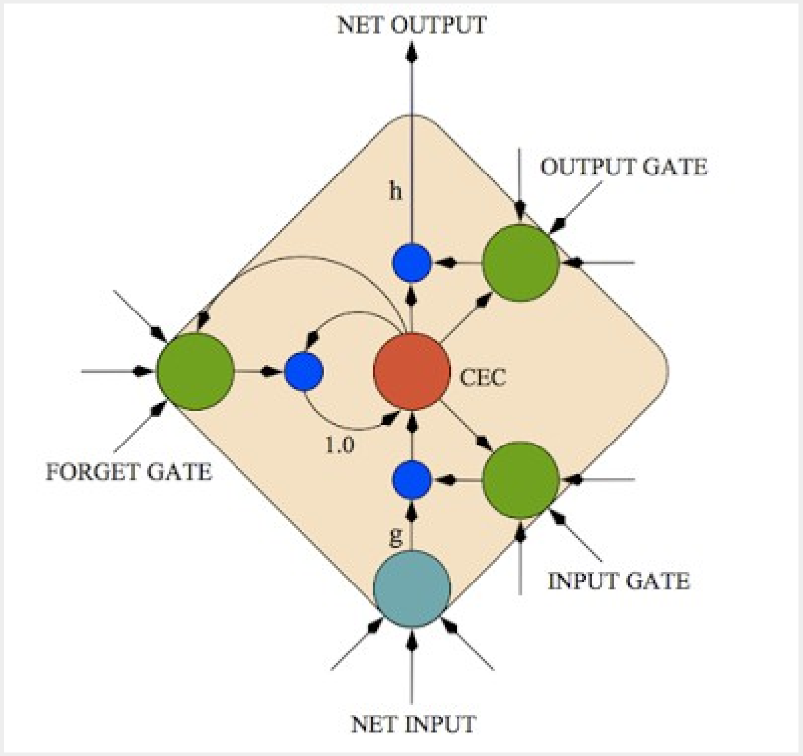
- Hopfield networks can act as error correction or vector categorisation (classifier) networks
- Input vectors are used as the initial conditions to the network, which recurrently updates until it reaches a stable output vector
- Hopfield networks are interesting from a theoretical standpoint, but are seldom used in practice
- Even the best Hopfield designs may have spurious stable points that lead to incorrect answers
- More efficient and reliable methods for these applications are available

Boltzmann Machines (very briefly)

- Boltzmann machines (BMs) are bidirectionally connected networks of stochastic processing units, which can be interpreted as neural network models
- can be seen as the stochastic, generative counterpart of Hopfield nets
- named after the Boltzmann distribution in statistical mechanics, which is used in their sampling function
- can be used to learn important aspects of an unknown probability distribution based on samples from this distribution
- In general, this learning process is difficult and time-consuming
- However, the learning problem can be simplified by imposing restrictions on the network topology, which leads us to restricted Boltzmann machines (RBMs)
- RBMs attracted much attention in recent years after being proposed as building blocks of multi-layer learning systems called ***deep belief networks***.

LSTM Networks

- Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture proposed by Hochreiter and Schmidhuber (1997)
- based on analysis of the problems that conventional learning algorithms, e.g. back-propagation through time (BPTT) have when learning timeseries with long-term dependencies – errors propagated back in time tend to either vanish or blow up
- LSTM's solution to this problem is to enforce constant error flow in specialized units, called Constant Error Carrousels (CECs)
- CECs have linear activation functions which do not decay over time
- well-suited to learn from experience to classify, process and predict time series when there are time lags of unknown size and bound between important events
- Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs and hidden Markov models and other sequence learning methods in numerous applications
- LSTM achieved the best known results in natural language text compression, unsegmented connected handwriting recognition, and in 2009 won the ICDAR handwriting competition.
- have been used for automatic speech recognition: a major component of network that in 2013 achieved a record 17.7% phoneme error rate on the classic TIMIT natural speech dataset
- As of 2016, major technology companies including Google, Apple, Microsoft are using LSTM networks as fundamental components in new products



Summary

- Recursive Neural Networks (e.g. Elman and Jordan networks), by having an internal feedback loop, are capable of learning to detect and generate temporal patterns.
- This makes them useful in such areas as signal processing and prediction where time plays a dominant role.
- Because they are an extension of the two-layer sigmoid/linear architecture, they inherit the ability to fit any input/output function with a finite number of discontinuities.
- They are also able to fit temporal patterns, but may need many neurons in the recurrent layer to fit a complex function.
- Hopfield networks also store temporal information but are of limited practical application
- Boltzman machines can be seen as stochastic counterpart of Hopfield networks
- Recent developments such as LSTM are now state-of-the art in speech recognition

References and Further Reading

- Jordan, M. I. (1986). “Serial order: A parallel distributed approach”. ICS Report 8604 San Diego, CA: University of California, Institute for Cognitive Science
- Elman, J.L. (1990). "Finding Structure in Time". Cognitive Science 14 (2): 179–211. doi:10.1016/0364-0213(90)90002-E
- Hopfield, J. (1982), “Neural Networks and Physical Systems with Emergent Collective Computational Abilities”, Proceedings of the National Academy of Sciences, Vol. 79, pp. 2554–2558.
- Hopfield, J., and D. Tank (1985), “Neural Computations of Decisions in Optimization Problems”, Biological Cybernetics, Vol. 52, pp. 141–152.
- Hinton, G.E., T.J.Sejnowski, D.Ackley (1984), “Boltzmann machines: constraint satisfaction networks that learn”. Technical Report CMU-CS-84–119, Carnegie-Mellon University
- Hochreiter,S, Schmidhuber, J (1997) “Long Short-Term Memory" Neural Computation, Vol. 9, No. 8, (doi: 10.1162/neco.1997.9.8.1735)

- Chapters 13 and 14 of Raúl Rojas. (1996) Neural Networks. A Systematic Introduction. Springer. Berlin Heidelberg NewYork. Available online at

  https://page.mi.fu-berlin.de/rojas/neural/neuron.pdf
- S. Haykin, Neural Networks,1999, Prentice Hall (chapter 15)
- D. Kriesel – A Brief Introduction to Neural Networks (ZETA2-EN). Available from:

  http://www.dkriesel.com/en/science/neural_networks