27-642/SDE 733
Artificial Neural Networks
Other ANN Models

Dr. Deborah A. Stacey

January, 1993

The Discrete Autocorrelator (DA)

Introduction

- a discrete autocorrelator (DA) is a **single-layer**, **symmetric**, **non-linear**, **autoassociative**, **nearest-neighbour** pattern encoder that stores binary/bipolar spatial patterns

$$A_k = (a_1^k, \ldots, a_n^k), \quad k = 1, 2, \ldots, m$$

  using Hebbian learning.

- the DA learns offline, asynchronously updates its PEs, operates in discrete time.

- also called the Hopfield Associative Memory or Hopfield Net.

Encoding

First-Order Encoding

- the encoding equation is:

$$W = \sum_{k=1}^{m} A_k^T A_k$$

  where W is the n-by-n matrix of F_A PE interconnections, or

$$w_{ij} = \sum_{k=1}^{m} a_i^k a_j^k$$

  for all i and $j = 1, 2, \ldots, n$, where w_{ij} is the strength of the symmetric connection strength ($w_{ij} = w_{ji}$) from the i^{th} to the j^{th} F_A PE.

- in Hopfield's net, there is the further restriction that the diagonal terms of W must be zero.

Higher-Order Encoding

- first-order connections correlate one PE's activation with another's.
  - can develop only linearly separable mappings.

- second-order connections correlate a pair of PE activations with another's.
  - can capture nonlinear mappings.
  - second-order correlations are stored in an n-by-n-by-n matrix V using:

$$v_{hij} = \sum_{k=1}^{m} a_h^k a_i^k a_j^k$$

for all h, i and $j = 1, 2, \ldots, n$, where v_{hij} is the strength of the symmetric connection strength from the h^{th} and i^{th} F_A PEs to the j^{th} F_A PE.

Recall

- the recall equation for the first-order DA is

$$a_i(t+1) = f(\sum_{j=1}^{n} w_{ij} a_j(t))$$

where $a_j(t)$ is the activation of the j^{th} PEs at time t and $f()$ is the binary/bipolar step threshold function

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- recall is a feedback operation that requires repeated application of the recall equation until all F_A PEs cease to change – i.e. the system is stable.
- the recall equation for the second-order DA is

$$a_j(t+1) = f(\sum_{h=1}^{n} \sum_{i=1}^{n} v_{hij} a_h(t) a_i(t))$$

Stability and Capacity

- it has been shown that the probability of perfect recall of all stored patterns is 1, provided the memory capacity does not exceed a logarithmic function that decreases for increasing n.

$$lim_{n \to \infty} P_n(m) = 1$$

provided the capacity does not exceed

$$m = \frac{n}{2 \ log \ n + log \ log \ n}$$

The Statistical Mechanics Approach

- some neural network models have a direct analogy with spin glass models of theoretical solid state physics.
- this analogy allows study of the properties of neural networks:
  1. storage capacity
     - calculation of the optimal storage capacity in the space of connection strengths
  2. quality of retrieval
  3. effects of noise

Spin Glasses

- magnetic materials which have a random orientational ordering (glass) of magnetic moments (spins).
- the spin sites are randomly interconnected by positive and negative competing interactions.
- spin glass dynamics are governed by a phase space with a large number of attractors.

Neural Nets and Statistical Mechanics

- for symmetrical interactions, it can be shown that the system evolves towards the minimum of an energy function.
- in the Hopfield model, the free energy of the system can be expressed as a function of the overlaps – the overlaps determine the closeness between the configuration of the network and the stored patterns during retrieval.
- the overlaps satisfy a set of self-consistency equations, from which statistical properties of the network are easily derived.

The Hopfield Net

- content addressable memory.
- aim: to recognize a partial or distorted input pattern, or code state vector, as being one of its previously memorized state vectors and to output the perfect and complete memorized version.
- a 'one-shot' weight setting scheme rather like the outer product or Hebbian learning rule.
- has been shown that the synchronous deterministic net behaves similarly to the original asynchronous stochastic network.

- consists of a network of N nodes with each node connected to the other nodes through a (synaptic) connection strength matrix T_{ij}.
- the nodes are simple processing elements with two possible outputs 1 and 0.
- represent the g^{th} pattern to be stored as a vector $g_i^{(s)}$ of length N with components 1 and 0.
- let $V_i \in \{0, 1\}$ denote the state of the neuron i.
- typical dynamical rules in neural networks belong to a general class of rule of the form:

$$V_i(t+1) = sgn(\sum_{j=1}^{N} T_{ij} V_j(t))$$

  where $sgn(x) = 1$ if $x > 0$ and $sgn(x) = 0$ if $x < 0$.
- the quantity:

$$\phi_i = \sum_{j=1}^{N} T_{ij} V_j$$

  corresponds to the biological membrane potential – the effect of the dynamical rule is to cause alignment between this potential vector and the next state vector (i.e. $V_i(t+1)\phi_i(t) \geq 0$).
- stable persisting states act as attractors – a stable long-term persisting pattern would satisfy:

$$V_i(t = \infty) = sgn(\sum_{j=1}^{N} T_{ij} V_j(t = \infty))$$

- if the nodes update asynchronously and the connection strengths are symmetric then only fixed point attractors occur (in the absence of noise).
- the change in weight is given by

$$\Delta T_{ij} = \begin{cases} V_i V_j & i \neq j \\ 0 & \text{otherwise} \end{cases}$$

- Hopfield's procedure operates by installing the memory structures as minima of the energy function:

$$E = -\frac{1}{2} \sum_{i \neq j} \sum_{i \neq j} T_{ij} V_i V_j$$

- one installed, each memory can be accessed from partial information by an iterative retrieval scheme which is a form of gradient descent in the energy function.
- an interesting property is that the contribution of a change in a unit, from 0 to 1 or vice-versa, to the total energy can be computed locally for each unit.
- the units are updated randomly and asynchronously by computing a potential change in energy (ΔE due to ΔV_i):

$$\Delta E = -\Delta V_i \sum_{j \neq i} T_{ij} V_j$$

Storage Capacity

- the Hopfield model, with the Hebbian rule, is relatively inefficient in terms of its storage capacity.
- it will only store a maximum of about $p = 0.145N$ uncorrelated patterns on an N-node network.
- it has been shown that the optimal storage capacity for an N-node network is $2N$ for uncorrelated patterns and more if the patterns are correlated.
- the storage capacity is limited by the noise generated by random overlaps between patterns.

Neurobiological Shortcomings

1. all the nodes are fully connected
    - in the cortex, a typical neuron is only connected to about $O(10^4)$ of the $O(10^{10})$ neurons present.
2. the Hebbian rule leads to symmetric T_{ij} connection strength matrices, so that node i influences node j in exactly the same way that j influences i
    - biological neurons are asymmetric pathways.
3. the learning rules are inadequate models of synaptic processes, since the matrix T_{ij} is unbounded and can have arbitrarily large values
    - only a restricted number of vesicles of neurotransmitter molecules are discharged at synaptic junctions, indicating that the synaptic efficacy cannot have a broad spectrum of values.

Problems

1. the larger the number of F_A PEs, the worse the storage capacity.
2. they only work effectively in conditions where the storage patterns are orthogonal to one another
    - Hopfield's solution: use vectors of high dimension, but this gives the network a very limited storage capacity of approximately 0.15 bits per weight.
3. if too many patterns are installed, there will be spurious minima in which the system can get stuck – since the system can only descend in energy, it will never be able to get out of unwanted minima.
    - set the initial state of the system a shorter Hamming distance away from the desired minimum than from any other minima. However, a means to ensure the correct starting state was not provided.
4. the restriction to only binary patterns.

Strengths

1. the ability to reconstruct entire patterns from partial or incomplete input – making it useful in noisy environments
    - well suited for applications that require the capability to remove noise from large binary patterns.
2. stability under asynchronous updating – making it appealing for integrated circuit implementations.
3. fault-tolerance.
4. applications include pattern matching and classification.

Circuit Realizations

- a research group at AT&T have developed a circuit modeled on the learning functions of the Limax garden slug.
    - slugs have very large neurons that can be easily studied.
    - they have only $20,000$ neurons - few enough to be catalogued.
- a model has been built that incorporates some of the lessons learned from studying Limax, both in the natural state and in computer simulations.
- the simplified model uses four neurons to represent the slug's taste buds that are connected to an autoassociative memory similar to the Hopfield net.
- the memory grid make associations by coasting to an energy minimum – a stable state the represents a solution to the problem that the slug faces.
- once a minimum has been reached, the circuit sends signals to the motor neurons – to flee or to eat (that is the question).

Hopfield's Chips

- the first standard problem tackled by Hopfield and Tank at Bell Labs was an analog-to-digital signal converter – demonstrated that a general method for emulating neurons can yield solutions to a variety of standard problems.
- only inherent limit for this design method is the power dissipation cost exacted by the resistive interconnections.

Reference Material

1. Simpson, Patrick K., **Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations**, 1990.
   - Chapter 5: ANS Paradigms and Their Applications and Implementations, pages 46-53.
2. Anderson, James A. and Edward Rosenfeld, editors, **Neurocomputing: Foundations of Research**, 1988.
   - "Neural networks and physical systems with emergent collective computational abilities", (1982) by J.J. Hopfield, pages 457-464.
3. Aleksander, Igor, editor, **Neural Computing Architectures: The Design of Brain-like Machines**, 1989.
   - Chapter 12: Statistical mechanics and neural networks by C. Campbell, D. Sherrington and K.Y.M. Wong, pages 239-257.
   - Chapter 6: A PDP learning approach to natural language understanding by N.E. Sharkey, pages 92-116.
4. Johnson, R. Colin and Chappell Brown, **Cognizers: Neural Networks and Machines that Think**, 1988.

The Discrete Autocorrelator Supplement

Key Contributions of Hopfield's Work

- conceptualized the network in terms of energy.
- showed that this is isomorphic to an Ising model (spin glasses).
- showed that an energy function exists for this network.
- processing elements with bi-stable outputs are guaranteed to converge to a stable local energy minimum
  - may consist of a stable oscillating series providing that each state has the same "energy" as the previous one.

Global Stability

- equilibria and motions that are only slightly perturbed by negligible deviations are called **stable**.
- globally stable ANSs are nonlinear dynamical systems that rapidly map all inputs to fixed points where information can be deliberately stored
  - all inputs are guaranteed to map to a fixed point although this might **not** be the desired fixed point.

Asynchronous Updating during Recall

- asynchronous updating is more "natural" than synchronous
  - no need for a clock.
- there are two different ways to approach this type of updating:
  1. at each time step, select at random a unit i to be updated, and apply the rule.
  2. let each unit independently choose to update itself with some constant probability per unit time.
- the first is appropriate for simulation and the second for autonomous hardware units.

Length of the Recall Cycle

- must specify how long the network will be allowed to cycle before recovering the stored pattern.
- we can demand that the network settle into a stable configuration.
- if a stable configuration is not reached after a prespecified length of time or number of interations, then the recall cycle can be terminated and the pattern recovered.

The Energy Function

- an important concept is to conceptualize the recall phase as an **energy function**

$$E = -\frac{1}{2}\sum_{ij} w_{ij} a_i a_j$$

- if we think of the space of all possible states of the network as the **configuraion space** (represented by a "region" then the stored patterns can be thought of as **attractors**.
  - we can then imagine an **energy landscape** "above" this configuration space.
- the central property of an energy function is that it is **always** decreasing (or remaining constant) as the system evolves according to its dynamical rule.
- thus the attractors (memorized patterns) are at local minima of the energy surface.
- the term **energy function** comes from a physical analogy to magnetic systems.
- for neural networks, an energy function exists if the connection strengths are **symmetric** $w_{ij} = w_{ji}$.
- Hebb's rule automatically yields symmetric w_{ij}'s.
- Hopfield's use of symmetric connections has been called a "clever step backwards from biological realism."
- many fields have a state function that is always decreasing during dynamical evolution or that must be minimized to find a stable or optimum state
  - theory of dynamical systems: **Lyapunov function**
  - statistical mechanics: **Hamiltonian function**
  - optimization theory: **cost or objective function**
  - evolutionary biology: **fitness function**

The Self-Organizing Map

Introduction

Brain Maps

- a detailed topological organization of the brain (cerebral cortex) can be deduced from functional deficits and behavioural impairments induced by various kinds of lesions, hemorrhages, tumours or malfunctions.
- different regions of the brain seem to be dedicated to specific tasks (localization of brain function).
- the various cortices in the cell mass seem to contain many kinds of "maps", such that a particular location of the neural response in the map often directly corresponds to a specific modality and quality of sensory signal.
- examples of brain maps
  1. primary visual cortex
     (a) field of vision "quasiconformal" map
     (b) line orientation map
     (c) colour map
  2. auditory cortex
     (a) tonotopic maps
  3. other sensory maps
     (a) somatotopic map of the skin's surface
  4. motor map
     (a) almost topologically identical to the somatotopic map
  5. higher level maps
     – usually unordered or kind of ultrametric topological order that is not easily interpreted.
     – also single cells that respond to rather complex patterns.
     – abstract qualities of sensory and other experiences

    - word processing – maps of categories and semantic values of words.

- possible conclusion: the internal representations of information in the brain are generally organized spatially.

The Self-Organization Model

- Example Problem
  - learning to approximate a mapping for robotic control strategy based only on randomly sampled inputs from a two-dimensional space.
- Example Scenerio
  - a robotic arm moving randomly in a 2-D plane.
  - arm is a line segment that can change its angle and length.
  - as the arm moves, its end effector is located to a random position in the x-y plane.
- The NN Model
  - a randomly selected coordinate pair form the arm's location sensor is fed as input to the net – the connection that becomes the most active is considered the image of the particular location.
  - this connection to a particular neuron defines the topological neighbourhood.
  - once the neighbourhood is defined, all the enclosed neurons change their connection strengths by an amount proportional to their distance from the input.
  - after the robot arm has sampled a sufficient number of uniformly distributed coordinate pairs, the net's connections become ordered according to their mutual similarity.
  - a particular neuron becomes sensitive to a particular connection that in turn becomes more active in response to a particular randomly sampled coordinate, forming an image of the 2-D space.
  - as more points are sampled, the lattice of lines begins to show the emergence of the topology-conserving property of the connections.
  - it is possible now to plot coordinate pairs that are totally different from any of the pairs that were used in forming the map – they are generalized as a nearest-neighbour match.
  - self-organization lets systems adapt to unpredictable changes in their environment – the net learns directly from its environment so that no extra constraints are required – allows functionality independent of any knowledge of the physical parameters of a particular system.
- Neural-Based Robotic Control
  - self-organization can enable control with inaccurately known mechanical structures or even if the mechanical structure has changed from mechanical deformation from bending, sliding or recoil.

- enables systems to make up for inaccuracies in mechanical structures and inaccurate sensor readings as they naturally degrade.
- if a neuron's excitation represents the rate of contraction of particular "muscle groups," then given the coordinates of a previously unseen position, the neurons' excitations determine where in space the arm will end up – numerical stability is guaranteed.
- in combination with optimization could be useful for robotic-control path minimization and collision avoidance.

The Self-Organizing Map as a Neural Network Model

- a sheet-like artificial neural network, the cells become specifically tuned to various input signal patterns or classes of patterns through an unsupervised learning process.
- the locations of the responses tend to become ordered as if some meaningful coordinate system for different input features was being created over the net.
- each cell or local cell group acts like a separate decoder for the same input.
- the spatial segmentation of different responses and their organization into topologically related subsets results in a high degree of efficiency.
- appears to be a scalable NN paradigm although it also seems that practical applications favour hierarchical systems made up of many smaller maps.

Competitive Learning

- assume a sequence of statistical samples of a vector $x = x(t) \in R^n$, where t is time, and a set of variable reference vectors $\{m_i(t) : m_i \in R^n,\ i = 1, 2, \ldots, k\}$.
- assume that the $m_i(0)$ have been initialized.
- if $x(t)$ can be simultaneously compared with each $m_i(t)$ at each successive instant of time then the best-matching $m_i(t)$ is to be updated to match even more closely the current $x(t)$.
  - using distance measure $d(x, m_i)$, altering m_i must be such that $d(x, m_c)$ is decreased, where m_c is the best-matching reference vector, and all other m_i are left intact.

Vector Quantization (VQ)

- produces an approximation to a continuous probability density function $p(x)$ of the vectorial input x using a finite number of "codebook" vectors, m_i.
- once the codebook is chosen, the approximation of x involves finding the reference vector m_c closest to x.

- one optimal placement of m_i minimizes E, the expected r^{th} power of the reconstruction error:

$$E = \int ||x - m_c||^r p(x) dx$$

  where the index c is a function of the input x

$$||x - m_c|| = MIN_i\{||x - m_i||\}$$

- there is no closed-form solution – iterative approximation schemes must be used.
- the steepest-descent gradient-step optimization of E in the m_c space yields the sequence:

$$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$$

$$m_i(t+1) = m_i(t) \quad for \quad i \neq c$$

  where $\alpha(t)$ is a suitable, monotonically decreasing sequence of scalar valued gain coefficients, $0 \leq \alpha(t) < 1$.
- the $m_i(t)$ develop into a set of feature-sensitive detectors.

The Self-Organizing Map Algorithm

- there are two essential effects leading to spatially organized maps:
  1. spatial concentration of the net activity on the cell (or its neighbourhood) that is best tuned to the present input.
  2. further sensitization or tuning of the best-matching cell and its topological neighbours to the present input.
- Selection of the Best-Matching Cell
  - let $x = [x_1, x_2, \ldots, x_n]^T \in R^n$ be the input vector that is connected in parallel to all the neurons i in the net.
  - the weight vector of cell i is denoted by $m_i = [m_{i1}, \ldots, m_{in}]^T \in R^n$.
  - measures for the match:
    - inner product $x^T m_i$.
    - Euclidean distances between x and m_i.
  - Note: the definition of x is only possible if the interrelation between the signals is simple,
    - e.g. in image analysis – preprocessing is needed to extract a set of invariant features for the components of x.
- Adaptation of the Weight Vectors
  - important that the cells doing the learning are not affected independently of each other but as topologically related subsets.

- weight vectors tend to attain values that are ordered along the axes of the network.
- to enforce lateral interaction, a neighbourhood set N_c is defined around cell c – at each learning step, all cells in N_c are updated; cells outside N_c are left intact.
- the radius of N_c can be time-variable
  * advantageous to let N_c be wide at the start and shrink monotonically with time.
  * may even end with $N_c = \{c\}$ or simple competitive learning.
- the updating process

$$m_i(t+1) = \begin{cases} m_i(t) + \alpha(t)[x(t) - m_i(t)] & \text{if } i \in N_c(t) \\ m_i(t) & \text{if } i \ni N_c(t) \end{cases}$$

  where $\alpha(t)$ is a scalar-valued adaptation gain, $0 < \alpha(t) < 1$. α should decrease with time.

General Comments on Map Algorithm

- in practical applications, the input and output weight vectors are usually of a high-dimension, e.g. in speech recognition it may be 15 to 100.
- input x is usually a random variable with a density function $p(x)$ from which successive values $x(t)$ are drawn, e.g. successive samples of input observables in their natural order of occurrence in speech recognition.
- the process starts with arbitrary or random initial values for the $m_i(0)$ – restricted to being different.

General Conditions on Map Algorithm

1. since learning is a stochastic process, the final statistical accuracy of the mapping depends on the number of steps, which must be reasonably large.
   - at least 500 times the number of net units.
   - the number of components in x has no effect – high dimensionality of input is allowed.
2. for approximately 1000 steps, $\alpha(t)$ should start with a value that is close to unity, thereafter decreasing monotonically – the ordering of m_i occurs during this initial period – remaining steps are only needed for refinement of the map.
   - after ordering, $\alpha(t)$ should retain small values over a long period.
3. the choice of N_c is critical – if the neighbourhood is too small to start with, the map will not be ordered globally – avoidable by starting with a fairly wide $N_c = N_c(0)$ and letting it shrink with time.

Learning Vector Quantization

- if the self-organizing map is to be used as a pattern classifier – problem becomes a decision process.
- the original map, like any VQ method is mainly intended to approximate input signal values or their probability density function by quantizing codebook vectors that are localized in the input space to minimize a quantization error functional.
- if the signal sets are to be classified into a finite number of categories then several codebook vectors are usually made to represent each class and their identity within the classes is no longer important – only decisions made at boundaries count.
- it is possible to define effective values for the codebook vectors such that they directly define non-optimal decision borders between the classes, even in the sense of classical Bayesian decision theory.
- recall closely coincides with that of the Bayes classifier.

General Comments on LVQ

- introduced by Kohonen in 1981.
- autoassociative, nearest-neighbour classifier.
- classifies arbitrary analog spatial patterns $A_k = (a_1^k, \ldots, a_n^k)$ where $k = 1, 2, \ldots, m$, into one of p-many classes using an error-correction encoding procedure.
- learns offline and in discrete time.
- has a two-layer topology where n F_A PEs correspond to A_k's components and p F_B PEs each represent a pattern class.
- because each F_B PE represents a class, these PEs are often referred to as "grandmother" cells.
- the learning is unsupervised and the recall is feedforward.
- classifies patterns by finding the optimal set of reference vectors (a set of connection that abut a F_B PE) for a given training set.
- the reference vectors are stored in the weight matrix W.
- during operation, the F_B PEs employ an invisible on-centre/off-surround competition that is used to choose the proper class for the presented input.
  - this includes lateral interactions.
  - self-exciting/neighbour-inhibiting connections.
- Topology

– fully-connected between F_A and F_B layers.

– negative lateral connections for each F_B PE to every other F_B PE.

– a positive recurrent connection from every F_B PE to itself.

Encoding

Single Winner Unsupervised Learning

- automatically determines the p-best reference vectors needed to represent the space spanned by a given set of data vectors.
- only allows one F_B PE to be active during the encoding phase – only the connections that abut the active F_B PE are adjusted.
- Procedure

1. Initialize all the F_A to F_B connection strengths to some random value in the range $[0, 1]$.
2. The connections that emanate from F_A and abut the j^{th} F_B PE form the weight vector W_j. For each pattern A_k do

   (a) find the W_j closest to A_k

   $$||A_k - W_g|| = MIN_{j=1}^{p}||A_k - W_j||$$

   where W_g is the W_j closest to A_k and the Euclidean distance between any two real-valued n-dimensional vectors X and Y is defined as

   $$||X - Y|| = [\sum_{i=1}^{n}(x_i - y_i)^2]^{\frac{1}{2}}$$

   This is a competition amongst the F_B PEs with the largest activation (closest reference vector to the data vector) remaining the winner.

   (b) move W_g closer to A_k using

   $$\Delta w_{ig} = \alpha(t)[a_i^k - w_{ig}]$$

   for all $i = 1, 2, \ldots, n$ where Δw_{ig} is the connection strength from the i^{th} F_A PE to the g^{th} F_B PE and $\alpha(t)$ is the learning rate at time t defined as

   $$\alpha(t) = t^{-1}$$

   or

   $$\alpha(t) = .2[1 - \frac{t}{10000}]$$

3. Repeat step 2 for $t = 1, 2, \ldots, z$ where $500 \leq z \leq 10,000$.

Multiple Winner Unsupervised Learning

- an extension allows more than one F_B PE to be active during encoding, thus allowing connections to each active F_B PE to be adjusted during training.
- introduce a set of F_B PEs, N_c, of a predetermined size in the physical neighbourhood of the winning F_B PE, W_g.
- adjust W_g and all the connections that abut the PEs in the neighbourhood of b_g.
- e.g. a set of three around winner W_g would be $N_g = \{W_{g-1}, W_g, W_{g+1}\}$.
- often the F_B PEs are arranged in a 2-D topology and the neighbourhood is a circle of predetermined radius around b_g.
- replace Δw_{ig} in the original procedure with

$$\Delta w_{ij} = \begin{cases} \alpha(t)[a_i^k - w_{ij}] & \text{if } j \in N_g \\ 0 & \text{otherwise} \end{cases}$$

Supervised Learning

- possible extension when the proper class for each A_k is known a priori.
- accelerates the learning and develops more accurate pattern classifications.
- supervision is by means of rewarding corrections for proper classification and punishing corrections for improper classifications.
- replace the Δw_{ig} equation with

$$\Delta w_{ig} = \begin{cases} +\alpha(t)[a_i^k - w_{ig}] & \text{if } b_g \text{ is the proper class} \\ -\alpha(t)[a_i^k - w_{ig}] & \text{if } b_g \text{ is not the proper class} \end{cases}$$

  where b_g is the F_B PE class chosen for A_k.

Recall

- determines the class, b_g, that the input pattern A is most closely associated with.
- W_g is determined by finding the closest W_j (Euclidean distance) to A.
- the F_B PEs all compete and the largest F_B PE activation prevails.
- at the end of the competition, the F_B PE representing the proper class will have a value of 1, all others will be 0.

$$b_j = \begin{cases} 1 & \text{if } ||A - W_g|| = MIN_{j=1}^{p} ||A - W_j|| \\ 0 & \text{otherwise} \end{cases}$$

Convergence

- determined by the learning parameter $\alpha(t)$ where $\alpha(t) \to 0 \ \ as \ \ t \to \infty$.
- analysis of the self-organizing process – after encoding is complete, each W_j represents the centroid of a decision region created in the n-dimensional space of data patterns.

Strengths

- ability to perform non-parametric pattern classification and provide real-time nearest-neighbour responses.
- ability to allocate reference vectors to the centroids of decision regions without any a priori information concerning data distributions.
- well suited to applications that require data quantization, e.g. statistical analysis, codebook communication, data compression, combinatorial optimization problems.
- one of the primary ANS paradigms because of its relative simplicity and its unsupervised learning.

Limitations

- extensive offline encoding time.
- inability to add new classes without complete retraining.

Extensions and Studies

- Desieno (1988) allows a "conscience" mechanism in each F_B PE that limits the number of times that it can win – provides much better pattern classification performance.
- Abutaleh (1988) has shown that the LVQ algorithm is a special case of an adaptive time-varying filter.
- Lutterell's (1988) extension allows the growth of the F_B layer to properly perform cluster decomposition problems.
- Kohonen (1988) has extended LVQ to develop more precise decision boundaries during self-organization.

Semantic Map Example

- the deepest semantic elements of any language should be physiologically represented in the neural realms.
- a NN for linguistic representations – difficult to find a metric distance between symbolic items.

  - the symbol during the learning process is presented in context.
  - similarity between items could be reflected through similarity of the contexts.

- let vector x_s represent the symbolic expression of an item.
- let vector x_c represent its context.
- input vector x is a concatenation of x_s and x_c – the norm of the context part predominates over that of the symbol part during the self-organizing process – the topological mapping mainly reflects the metric relationship of the sets of associated encodings.
- the symbols become encoded into a spatial order reflecting their logical (or semantic) similarities.
- use a simple language
  - vocabulary – nouns, verbs, adverbs.
  - a sequence of randomly generated meaningful 3-word sentences as input $\rightarrow$ concatenated into a single continuous string, S.
  - context – the word before and after – words from other sentences are uncorrelated and act as random noise.
- code vectors of the predecessor/successor pair forming the context are concatenated into a single 14-D code vector X_c.
- first consider each word in its average context – defined as the average over 10,000 sentences of all code vectors of predecessor/successor pairs surrounding a particular word.
- results in 30 14-D "average word contexts" – assuming role of context fields X_c which is combined with X_s – the symbol.
- used rectangular lattice of 10 by 15 cells.
- after 2000 input presentations, the responses of the neurons to presentation of the symbol parts alone were tested.
- symbolic label is written to the site which gave the maximum response.
- the contexts "channel" the word items to memory positions whose arrangement reflects both grammatical and semantic relationships.

Practical Applications of the Map

- statistical pattern recognition, especially recognition of speech.
- control of robot arms and other problems in robotics.
- control of industrial processes.

- automatic synthesis of digital systems.
- adaptive devices for various telecommunications tasks.
- image compression.
- radar classification of sea-ice.
- optimization problems.
- sentence understanding.
- application of expertise in conceptual domains.
- classification of insect courtship songs.

The Difference between the Self-Organizing Map and other Neural Network Paradigms

- most ANSs strongly emphasize distributed representations and only consider spatial organization of the PEs as a secondary aspect.
- the intrinsic potential of the self-organizing map process for creating a localized, structured arrangement of representations in the basic net module is emphasized.
- the massive interconnects that underlie all neural processes are certainly distributed but their effects may be focused on local sites.
- any complex processing task requires organization of information into separate parts.

References

1. Simpson, Patrick K., Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations, 1990.
    - Chapter 5: pages 85-90.
2. Kohonen, Teuvo, "The Self-Organizing Map", Proceedings of the IEEE, Volume 78, Number 9, September, 1990, pp 1464-1480.
3. Josin, Gary, "Neural-Network Heuristics", Byte, October, 1987, pp 183-192.

The Self-Organizing Map Supplement

Self-Organizing Maps

- developed by Teuvo Kohonen between 1979 and 1982.
- creates a 2 dimensional feature map of the input data so that order is preserved.
- primary use is to visualize topologies and hierarchical structures of higher dimensional input spaces
- can be used in hybrid networks as a front-end to a supervised network.
- thought to act similarly to biological system
  - preserves order
  - compacts the representation of sparse data
  - spreads out dense data throughout a 2-D region
- in human brain:
  - "the map of acoustic frequencies on the auditory cortex is perfectly ordered and almost logarithmic with respect to frequency"

Algorithm to Produce a Self-Organizing Feature Map

Step 1. Initialize weights

- initialize weights from N inputs to the M output nodes to small random values
- set the initial radius of the neighbourhood

Step 2. Present new data

Step 3. Compute distances to all nodes

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2$$

Step 4. Select output node with minimum distance

- select node j^* with minimum d_j

Step 5. Update weights to node j^* and neighbours

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)(x_i(t) - w_{ij}(t))$$

$$for\ \ j \in N_{j^*}(t) \quad 0 \leq i \leq N-1 \quad 0 \leq \alpha(t) \leq 1 \quad where\ \ \alpha\ \ decreases\ \ with\ \ time$$

Step 6. Repeat by going to Step 2 until $\alpha(t) = 0$

The Plotting of Weight Vectors

- the following comments refer to the diagram of six subplots
- the weights for 100 output nodes are plotted in 6 subplots
  - 2 random independent inputs uniformly distributed over the region enclosed by the boxed area
- line intersections specify weights for one output node
  - weights from input x_0 are specified by the position along the horizontal axis and weights from input x_1 are specified by the position along the vertical axis
  - lines connect weight values for nodes that are topological nearest neighbours
  - weights start at time zero clustered at the centre of the plot
  - weights then gradually expand in an orderly fashion until their point density approximates the uniform distribution of the input samples
    - an orderly grid indicates that topologically close nodes code inputs that are physically similar

A Conscience Mechanism

- one problem of a *competitive* algorithm is that one node can end up representing too much of the input data – it "wins" too often
  - to solve this problem, a **conscience** mechanism was introduced by DeSieno in 1988
    - a record is kept of how often each node wins and this is used during training to "bias" the distance measure
    - if a node has won more than $\frac{1}{N}$ times where N is the number of output or "Kohonen" nodes than its distance is adjusted upwards to decrease its chance of winning – proportional to how much the average node has won

    * nodes that have won less than $\frac{1}{N}$ times then have their distances decreased to make winning more likely
  - the conscience mechanism helps nodes represent approximately equal information about the input data
    * sparse data is compacted
    * high density input is spread out to allow fine discrimination
      · thought to mimic the knowledge representation of biological systems

Concluding Remarks

- SOMs are viable sequential vector quantizers when the number of clusters desired can be specified before use and the amount of training data is relatively large compared to the number of clusters desired
  - similar to *K-means clustering*
  - results may depend on the presentation order of the input data for small amounts of training data

And Now for a Little Statistics and Probability...

Abduction

- **Abduction** is the generation of explanations for what we see around us.
- It could be thought of as having the following form:

$$\frac{\begin{array}{l}\text{(if a b)}\\ \text{b}\end{array}}{\text{a}}$$

- Example:

$$\frac{\begin{array}{l}\text{(if (drunk ?person)(not (walk-straight ?person)))}\\ \text{(not (walk-straight Jack))}\end{array}}{\text{(drunk Jack)}}$$

- Abduction is only **plausible inference**.

Abduction and Causation

- Not all inferences of the form "(if a b) and b, therefore a" can be thought of as generating an explanation.
- Example:

(if (in ?patient ward5)(have ?patient cancer))
(have Eliza cancer)

(in Eliza ward5)

- Most notions of **explanation** have a lot to do with **causality**.
- Causality and logical implication are not the same notions.
- General Form:

(cause ?x ?y)	**; ?x causes ?y**
?y	**; and ?y is true**
?x	**; so hypothesize ?x as explanation**

Abduction and Evidence

- Abduction requires that we find pertinent facts and apply them to infer a new fact.
- We can get more than one answer in abductive reasoning.
- There is no foolproof way to decide between alternatives.
- The best we can do is to find one hypothesis more likely, or **probable**.
- We must know
  1. how strongly a fact weighs for or against a conclusion, and
  2. how to combine pieces of evidence into a final conclusion.
- Statistics and probability theory offer well-understood ways of doing this.

Basic Definitions

- Consider a problem of medical diagnosis - a typical abduction problem.
- One approach - of all the people in the world who have, or had, the exact same symptoms as this patient, what did they suffer from?
- Consider a case about which one fact is known: the patient's skin is too yellow.
- **Conditional probability**: what is the likelihood of the patient suffering from a particular disease given this symptom.
- It is written:

$$P(disease \mid symptom)$$

- and read as:

the probability of disease given symptom

- Example: if 19% of all people who have overly yellow skin have damage to the liver from overconsumption of alcohol:

$$P(pickled - liver \mid yellow - skin) = 0.19$$

- **Unconditional probability** is the probability of the disease before any symptoms have been seen.
- It is written:

$$P(disease)$$

- Unconditional probabilities are also called **prior probabilities** because they are the probability of something **prior** to any evidence.

Back to Conditional Probability

$$P(pickled\ liver) = \frac{\mid Pickled - Liver \mid}{\mid People \mid}$$

$$P(p'ed\ liver \mid yellow\ skin) = \frac{\mid Pickled - Liver \cap Yellow - Skin \mid}{\mid Yellow - Skin \mid}$$

$$P(yellow\ skin \mid p'ed\ liver) = \frac{\mid Yellow - Skin \cap Pickled - Liver \mid}{\mid Pickled - Liver \mid}$$

In general, the patient has n symptoms, so for each disease d_i we want to know:

$$P(d_i \mid s_1 \& s_2 \& \ldots \& s_n) = \frac{\mid d \cap s_1 \cap s_2 \ldots \cap s_n \mid}{\mid s_1 \cap s_2 \ldots \cap s_n \mid}$$

Probabilities

- some information will be of the negative sort, e.g. the patient does not have yellow skin.

$$P(not\ \ s) = 1 - P(s)$$

$$P(not\ \ s \mid d) = 1 - P(s \mid d)$$

- we can represent conditional probabilities as facts about causal relations, e.g.

```
(cause   (condition-of (liver-of ?patient) pickled)
         (colour-of (skin-of ?patient) yellow))
```

- we could then add extra arguments indicating the statistical relations between the two:

```
(cause   (condition-of (liver-of ?patient) pickled)   disease
         (colour-of (skin-of ?patient) yellow)        symptom
         .50                                          P(s | d)
         .01)                                         P(d | s)
```

Baye's Theorem

- we cannot directly use the definition for the conditional probability of pickled liver given yellow skin because we never know

  "...of the i people with yellow skin, j had pickled liver"

- but we do know

  "...how many patients with a given disease show a particular symptom"

- example:

$$P(puffy\ \ eyelids \mid pickled\ \ liver) \sim 1\%$$
$$P(yellow\ \ skin \mid pickled\ \ liver) \sim 90\%$$

- Baye's theorem is a way to calculate conditional probabilities.

$$P(d \mid s) = \frac{P(d)P(s \mid d)}{P(s)}$$

$$\frac{|d \cap s|}{|s|} = \frac{|d|}{|PEOPLE|} \times \frac{\frac{|s \cap d|}{|d|}}{\frac{|s|}{|PEOPLE|}}$$

- the numbers on the RHS are easily available (compared to those on the LHS).

Single Disease / Single Symptom

- for the simple case of the conditional probabilities for a single disease given a single symptom
  - if we have m diseases and n symptoms then we need $n \times m$ numbers:

$$(mn\ \ conditional\ \ probabilities) + (m\ \ disease\ \ probabilities) +$$
$$(n\ \ symptom\ \ probabilities) = mn + m + n \approx mn$$

- not necessary to find all numbers - some diseases and symptoms are not related and therefore their conditional probabilities are close to zero.

Multiple Symptoms

- Baye's theorem for two symptoms:

$$P(d \mid s_i \& s_j) = \frac{P(d)P(s_i \& s_j \mid d)}{P(s_i \& s_j)}$$

- for every pair of symptoms we need $P(s_i \& s_j \mid d)$ and $P(s_i \& s_j)$
- the number of such pairs is $n \times (n-1) \approx n^2$

The Assumption of Statistical Independence

- the statistical independence of two symptoms means that the probability of seeing one given the other is exactly the same as the general probability of seeing the first,

$$P(s_i \mid s_j) = P(s_i)$$

- given the statistical independence of two symptoms, one of the numbers can be computed using

$$P(s_i \& s_j) = P(s_i)P(s_j)$$

- saying that a disease is "unrelated" to a symptom is really an informal way of saying that the disease is statistically independent of the symptom

$$P(d_m \mid s_j) = P(d_m)$$

that is, the probability of seeing the disease after seeing the symptom is exactly the same as before we saw it - the prior probability of the disease.

Another Assumption

- two symptoms are not only independent among people at large but also in the subset of people suffering from disease d

$$P(s_i \mid s_j \& d) = P(s_i \mid d)$$

$$\rightarrow P(s_i \& s_j \mid d) = P(s_i \mid d)P(s_j \mid d)$$

- by using the two independence assumptions in Baye's theorem:

$$P(d \mid s_i \& s_j) = \frac{P(d)P(s_i \mid d)P(s_j \mid d)}{P(s_i)P(s_j)}$$

- the numbers that this requires are exactly the same as in the case of only one symptom
- based on strong assumptions that our symptoms are really measuring different things, we eliminate the need for large amounts of statistical information

$$\begin{aligned} P(d \mid s_i \& s_j) &= P(d)\left[\frac{P(s_i|d)}{P(s_i)}\right]\left[\frac{P(s_j|d)}{P(s_j)}\right] \\ &= P(d)I(d \mid s_i)I(d \mid s_j) \end{aligned}$$

$$where \;\; I(d \mid s) = \frac{P(s \mid d)}{P(s)}$$

Discrete Bidirectional Associative Memory

- BAMs represent the logical extension of autocorrelators to heterocorrelators.
- with heterocorrelators, feedback between layers stabilizes and at the point of stability is a stored pattern.
- this idea was introduced in 1986 (Soffer, Marom, Owechko, and Dunning) for use in an optical resonating associative memory.
- in 1988, Kosko refined first-order heterocorrelators and called them bidirectional associative memories (BAMs).
- the BAM is a two-layer, heteroassociative, nearest-neighbour pattern matcher.
- it encodes arbitrary binary/bipolar spatial pattern pairs (A_k, B_k) using Hebbian learning, where the k^{th} pattern pair is represented by the vectors $A_k = (a_1^k, \ldots, a_n^k)$ and $B_k = (b_1^k, \ldots, b_p^k)$.
- the BAM has the ability to learn online and operates in discrete time.
- BAM is similar to the ART1 system.

Encoding

- encoding is performed by summing together the outer products of the m pattern pairs using the equation

$$W = \sum_{k=1}^{m} A_k^T B_k$$

or

$$w_{ij} = \sum_{k=1}^{m} a_i^k b_j^k$$

where w_{ij} is the connection strength from the i^{th} F_A to the j^{th} F_B PE.

Recall

- BAM recall employs inter-layer feedback between F_A and F_B PEs.
- given an initial pattern or two incomplete pattern pairs, the BAM immediately reaches a resonant state where the final pattern pairs are found.
- when BAM updates are synchronous then all the PEs from one layer are activated simultaneously.
- when BAM updates are asynchronous then less than all the PEs in one layer are activated at any given time.

The Recall Operation

1. Present a pattern to F_A or F_B or partials to both.
2. Feed the F_A activations asynchronously (or synchronously) through W to F_B.
3. Calculate the F_B activations.
4. Feed back the F_B activations asynchronously (or synchronously) through W^T to F_A.
5. Calculate the F_A activations.
6. Repeat steps 3 to 6 until all F_A and F_B activations cease to change. This is the state of equilibrium called **resonation**.

- the F_B activations are calculated by

$$b_j(t+1) = \begin{cases} 1 & \text{if } y_j > 0 \\ b_j(t) & \text{if } y_j = 0 \\ -1 & \text{if } y_j < 0 \end{cases}$$

where $b_j(t+1)$ is the activation of the j^{th} F_B PE at time $t+1$ and y_j is the j^{th} F_B PE's pre-activation value found by

$$y_j = \sum_{i=1}^{n} a_i(t) w_{ij}$$

where $a_i(t)$ is the activation of the i^{th} F_A PE at time t.

- the F_A activations are calculated by

$$a_i(t+1) = \begin{cases} 1 & \text{if } x_i > 0 \\ a_i(t) & \text{if } x_i = 0 \\ -1 & \text{if } x_i < 0 \end{cases}$$

where x_i is the i^{th} F_A PE's pre-activation value found by

$$x_i = \sum_{j=1}^{p} b_j(t) w_{ji}$$

where w_{ji} is the connection strength from the j^{th} F_B to the i^{th} F_A PE.

Stability

- Kosko proved BAM stability using a discrete-time Lyapunov energy function similar to the Hopfield net's.
-

$$L(A,B) = -\sum_{i=1}^{n}\sum_{j=1}^{p} a_i b_j w_{ij}$$

$$\Delta L_A(A,B) = -\sum_{i=1}^{n} \Delta a_i \sum_{j=1}^{p} b_j w_{ij}$$

$$\Delta L_B(A,B) = -\sum_{j=1}^{p} \Delta b_j \sum_{i=1}^{n} a_i w_{ij}$$

- since all possible non-zero PE activation changes lead to a decrease in energy and all zero PE activation changes result in equilibrium, the BAM is globally stable.
- since the weight matrix, W, has no effect on the stability analysis, all W are bidirectionally stable.

In Conclusion

BAM Strengths

- unconditional stability.
- instructive simplicity.
- ability to add new pattern pairs quickly.

BAM Limitations

- inability to encode a large number of pattern pairs
  - the BAM's capacity is

    $$m = \frac{q}{4\ log\ q}$$

    where $q = min(n,p)$.
- restriction to binary/bipolar valued pattern pairs.

Applications

- the BAM is best applied in situations that require nearest-neighbour pattern matching of only a few patterns.
- like the Hopfield Net, the BAM is extremely amenable to optical implementations because of its simple dynamics.

Bidirectional Associative Memory Supplement

Feedback Networks

- suppose n neurons in field F_X connect to p neurons in field F_Y
- a connection matrix M summarizes the synapses between the F_X and F_Y neurons
  - a $n - by - p$ matrix of real numbers, m_{ij}
  - describes the **forward projections** from F_X to F_Y
- the $p - by - n$ matrix N describes the **backward projections** from F_Y to F_X
- if $M = N^T$ and $N = M^T$, this defines the minimal two-layer feedback network or **bidirectional network**
- when the activation dynamics of F_X and F_Y lead to overall stable behaviour $\rightarrow$ **bidirectional associative memory** or **BAM**
- the simplest type of BAM network:
  - additive
  - two-valued
  - nonadaptive
- a good example of a feedback nonlinear neural network

Definition

$$x_i^{k+1} = \sum_j^p S_j(y_j^k) m_{ij} + I_i$$

$$y_j^{k+1} = \sum_i^n S_i(x_i^k) m_{ij} + J_j$$

- where M is an arbitrary but constant synaptic connection matrix using discrete time steps k and signal functions S_i and S_j which represent binary or bipolar threshold functions

Threshold Functions

$$S_i(x_i^k = \begin{cases} 1 & if \;\; x_i^k > U_i \\ S_i(x_i^{k-1}) & if \;\; x_i^k = U_i \\ 0 & if \;\; x_i^k < U_i \end{cases}$$

$$S_j(y_j^k) = \begin{cases} 1 & if \;\; y_j^k > V_j \\ S_j(y_j^{k-1}) & if \;\; y_j^k = V_j \\ 0 & if \;\; y_j^k < V_j \end{cases}$$

- for arbitrary real valued thresholds $U = (U_1, \ldots, U_n)$ for F_X neurons and $V = (V_1, \ldots, V_p)$ for F_Y
- the stay-the-same value occurs infrequently in large nets
- the bivalent signal functions allow the modelling of complex asynchronous state-change patterns
- each neuron may randomly decide whether to check the threshold conditions
- this randomness may be described with slowly varying means and variances of state-change frequency
  - the network behaves as a vector stochastic process
  - each neuron behaves as a scalar stochastic process
- state changes may be made **synchronously** – an entire field at a time
- the other extreme is **simple asynchrony** – only one neuron makes a state change decision at a time
- in general, **subset asynchronous** state changes are used

An Example

- the BAM model can be illustrated with any real valued matrix
- the BAM network converges to fixed points for every matrix with arbitrary subset asynchronous state-change policies
- example:
  - start with a BAM with 4 neurons in F_X and 3 neurons in F_Y

- assume that information is already encoded in the matrix M

$$M = \begin{pmatrix} -3 & 0 & 2 \\ 1 & -2 & 0 \\ 0 & 3 & 2 \\ -2 & 1 & -1 \end{pmatrix}$$

$$M^T = \begin{pmatrix} -3 & 1 & 0 & 2 \\ 0 & -2 & 3 & 1 \\ 2 & 0 & 2 & -1 \end{pmatrix}$$

- at time k, all F_Y neurons are ON; $S(Y_k) = (1 \;\; 1 \;\; 1)$
- suppose X_k at F_X is $(5 \;\; -2 \;\; 3 \;\; 1)$
- for simplicity, all thresholds are zero; $U_i = V_j = 0$ for all i, j
- using synchronous state-changes $\rightarrow$ at $k+1$, F_X neurons produce a binary signal state vector $S(X_k)$ which results in $S(X_k) = (1 \;\; 0 \;\; 1 \;\; 1)$
- at $k+1$, these F_X signals pass "forward" through M to affect the activations of the F_Y neurons
- the 3 F_Y neurons compute 3 dot products or correlations $\rightarrow$ $S(X_k)M = (-5 \;\; 4 \;\; 3) = Y_{k+1}$
- computing the new signal state vector gives $S(Y_{k+1}) = (0 \;\; 1 \;\; 1)$
- observations:
  - the first F_Y neuron has changed states from ON to OFF
  - an asynchronous state-change policy may not produce this state change
- the signal state vector $S(Y_{k+1})$ now passes "backwards" through the synaptic filter M^T at time $k+2$ to yield $S(Y_{k+1})M^T = (2 \;\; -2 \;\; 5 \;\; 0) = X_{k+2}$
- thresholding F_X at $k+2$ reveals a BAM fixed point equilibrium $S(X_{k+2}) = (1 \;\; 0 \;\; 1 \;\; 1) = S(X_k)$
- since $S(X_{k+2}) = S(X_k)$, passing $S(X_{k+2})$ through M will produce $S(Y_{k+1})$ at F_Y at time $k+3$
- passing $S(Y_{k+3}) = S(Y_{k+1})$ backwards through M^T will produce $S(X_{k+2})$ again at F_X – **bidirectional equilibrium**
- asynchronous state changes may lead to different bidirectional equilibria
- the system is guaranteed to reach an equilibrium
- all BAM state changes lead to fixed point stability
  - synchronous
  - asynchronous

Lyapunov Minimia

- BAM dynamical networks perform $O(1)$ search for stored patterns $\rightarrow$ depends on how information is encoded in M
- correlation encoding techniques tend to store information at local "energy" or Lyapunov minimia
- the time it takes to descend into an attractor basin does not depend on the number of basins/stored patterns
- this **dimension-independent** property holds for all globally stable neural networks
- familiar patterns produce "deep" attractor basins
- classification accuracy may change with time – overcrowding may degrade classification accuracy – similar learned patterns may have overlapping basins and spurious basins may occur
- order-one search provides a central advantage of dynamical systems computation relative to digital computation
  - with computers, search time increases as the number of memory items increases

BAM Connection Matrices

- any method of learning can produce M and M^T
- the most popular method is the **bipolar Hebbian** or **outer-product** learning method
  - sum of weighted correlation matrices
- can use binary or bipolar vectors and the Boolean sum can replace the arithmetic sum
- take m pairs of binary vectors (A_i, B_i) or bipolar vectors (X_i, Y_i)
- the bipolar outer-product law sums the individual $n-by-p$ bipolar correlation matrices $X_k^T Y_k$

$$M = \sum_k^m X_k^T Y_k$$

- we could also assign real valued weights w_k to the association (A_k, B_k) and then use the weighted outer-product law

$$M = \sum_k^m w_k X_k^T Y_k$$

- we could also arrange the weights to produce a **recency** or **fading-memory** effect – give more weight to more recent associations

  - **simple recency effect**: $w_1 < w_2 < \ldots < w_m$
  - **exponential fading memory**: $w_k = c^{m-k}$ where $0 < c < 1$

- an appropriately weighted fading memory network yields a "moving window" nonlinear filter for an arbitrary sequence of training pairs
- on average, bipolar signal state vectors produce more accurate recall than binary signal state vectors when bipolar outer-product encoding is used
- example: F_X has 6 neurons and F_Y has 4 neurons
- training pairs or associations
  - $X_1 = (1 \;\; -1 \;\; 1 \;\; -1 \;\; 1 \;\; -1) \quad Y_1 = (1 \;\; 1 \;\; -1 \;\; -1)$
  - $X_2 = (1 \;\; 1 \;\; 1 \;\; -1 \;\; -1 \;\; -1) \quad Y_2 = (1 \;\; -1 \;\; 1 \;\; -1)$
- BAM memory matrix M is constructed by adding the bipolar correlation matrices $X_1^T Y_1$ and $X_2^T Y_2$

$$X_1^T Y_1 = \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$$

$$X_2^T Y_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

$$M = X_1^T Y_1 + X_2^T Y_2 = \begin{pmatrix} 2 & 0 & 0 & 2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 0 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ -2 & 0 & 0 & 2 \end{pmatrix}$$

- assume all thresholds and inputs are zero and all update policies are synchronous

- present X_1 as input to the system – the current signal state vector at F_X

$$X_1 M = (8 \;\; 4 \;\; -4 \;\; -8) \rightarrow (1 \;\; 1 \;\; -1 \;\; -1) = Y_1$$

$$Y_1 M^T = (4 \;\; -4 \;\; 4 \;\; -4 \;\; 4 \;\; -4) \rightarrow (1 \;\; -1 \;\; 1 \;\; -1 \;\; 1 \;\; -1) = X_1$$

- so (X_1, Y_1) is a fixed point of the BAM dynamical system
- similarly

$$X_2 M = (8 \;\; -4 \;\; 4 \;\; -8) \rightarrow (1 \;\; -1 \;\; 1 \;\; -1) = Y_2$$

$$Y_2 M^T = (4 \;\; 4 \;\; 4 \;\; -4 \;\; -4 \;\; -4) \rightarrow (1 \;\; 1 \;\; 1 \;\; -1 \;\; -1 \;\; -1) = X_2$$

- for a 1-bit "noisy" input $X = (-1 \;\; 1 \;\; 1 \;\; -1 \;\; -1 \;\; -1)$

$$XM = (4 \;\; -4 \;\; 4 \;\; -4) \rightarrow (1 \;\; -1 \;\; 1 \;\; -1) = Y_2$$

$$Y_2 M^T = (4 \;\; 4 \;\; 4 \;\; -4 \;\; -4 \;\; -4) \rightarrow (1 \;\; 1 \;\; 1 \;\; -1 \;\; -1 \;\; -1) = X_2$$

- for a still "noisier" input $X = (-1 \;\; -1 \;\; -1 \;\; 1 \;\; 1 \;\; -1)$

$$XM = (-4 \;\; 4 \;\; -4 \;\; 4) \rightarrow (-1 \;\; 1 \;\; -1 \;\; 1) = -Y_2 = Y_2^c$$

$$Y_2^c M^T = (-4 \;\; -4 \;\; -4 \;\; 4 \;\; 4 \;\; 4) \rightarrow (-1 \;\; -1 \;\; -1 \;\; 1 \;\; 1 \;\; 1) = X_2^c$$

- this establishes the complement pair (X_2^c, Y_2^c) as another fixed point – a spurious attractor
  - for simple Hebbian correlation encoding, spurious attractors tend to increase in frequency as the network dimensionality increases
- BAMs work because the system dynamics guide the local behaviour to a global reconstruction (recollection) of a stored pattern
- the equilibrating system self-organizes its internal parameters in response to sampled stimuli

Capacity

- as we sum correlation matrices in a binary Boolean outer-product matrix, $m_{ij} = 1$ becomes more frequent until adding more associations does not significantly change the matrix – some patterns are "forgotten"
- the network tends to exceed its **memory capacity** (the number of distinct patterns that can be learned and recalled) as the number m of patterns approaches the number $min(n, p)$ of neurons in the network
- dimensionality limits capacity
- Grossberg's **Sparse Coding Theorem** (1976)

  - for deterministic encoding, pattern dimensionality must exceed pattern number to prevent learning some patterns at the expense of others

- for Boolean encoding of binary associations, memory capacity can greatly exceed $min(n, p)$ of the thresholds U_i and V_j are chosen carefully $\rightarrow$ new upper bound is $min(2^n, 2^p)$

  - heuristics and exhaustive search have been used to determine high capacity sets of thresholds

The Boltzmann Machine

Introduction

- the Boltzmann Machine is an example of a supervised learning network.
- it is one of the few algorithms which have extended the single layer perceptron learning algorithm to multiple layers.
- it is a simple extension of the Hopfield model.
- BM is a heteroassociative, nearest-neighbour pattern matcher that stores arbitrary binary spatial patterns using a combination of Hebbian encoding and simulated annealing.
- type of parallel constraint network that is capable of learning the underlying constraints that characterize a domain simply by being shown examples from the domain.
- the network modifies the strength of its connections so as to construct an internal generative model that produces examples with the same probability distribution as the examples it is shown.
- shown any particular example, the network can **interpret** it by finding values of variables in the internal model that would generate the example.
- when shown a partial example, the network can complete it by finding internal variable values that generate the partial example and using them to generate the remainder.
- BM is a parallel computational organization that is well suited to constraint satisfaction tasks involving large numbers of "weak" constraints.
- "strong" constraints **must** be satisfied by any solution – "weak" constraints incur a cost when violated – the quality of a solution is then determined by the total cost of all the constraints that it violates.
- the BM is composed of
    1. primitive computing elements called **units**,
    2. bidirectional links connecting the units.

- a unit is always **ON** or **OFF** based on a probabilistic function of the states of its neighbouring units and the **weights** on its links to them.
- a unit being ON means that the system currently accepts some elemental hypothesis about the domain.
- the weight on the link represents a weak pairwise constraint between two hypotheses. A positive weight indicates that the two hypotheses tend to support one another.
- link weights are **symmetric**.
- each global state of the network can be assigned a single number called the **energy** of that state.
- the individual units can be made to act so as to **minimize the global energy**.
- if some of the units are externally forced or **clamped** into particular states to represent input, the system will then find the minimum energy configuration that is compatible with that input.
  - i.e. the energy of a configuration can be interpreted as the extent to which the combination of hypotheses violates the constraints implicit in the problem domain, so in minimizing energy the system increasingly satisfies the constraints.
- the energy of a global configuration is defined as

$$E = -\sum_{i<j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

  where w_{ij} is the connection weight between i and j, s_i is 1 if unit i is on, and θ_i is a threshold.
- a simple algorithm for finding a combination of truth values that is a **local** minimum is to switch each hypothesis into whichever of its two states yields the lower total energy given the current state of the other hypotheses.
- if the units act **asynchronously** and if transmission times are **negligible**, then the system always settles into a local energy minimum.
- because the connections are symmetric, the difference in global energy with the k^{th} unit ON and the energy with it OFF can be determined locally by the k^{th} unit and this energy "gap" is

$$\Delta E_k = \sum_i w_{ki} s_i - \theta_k$$

Escaping Local Minima

- a weakness of gradient descent methods – it gets stuck in **local** minima that are not globally optimal.
- this is not a problem in Hopfield nets – the local energy minima are used to store "items".

- but for constraint satisfaction tasks, the system must try to escape from local minima in order to find the configuration that is the global minimum given the current input.
- a simple way to escape is to occasionally allow jumps to configurations of higher energy.
- an algorithm with this property is the **Metropolis algorithm**.
- the decision rule is the same as that for a particle which has two energy states.
- the system will reach **thermal equilibrium** and the relative probability of two global states will obey the Boltzmann distribution.
- this resembles the input-output function for a cortical neuron.

The Metropolis Algorithm

- Given that the **energy** of a global configuration is

$$E = -\sum_{i>j} w_{ij} s_i s_j + \sum_i \theta_i s_i$$

- If the **energy gap** is

$$\Delta E_k = \sum_i w_{ki} s_i - \theta_k$$

- Then regardless of the previous state, set $s_k = 1$ with probability

$$p_k = \frac{1}{(1 + e^{-\Delta E_k / T})}$$

  where T is a parameter that acts like temperature.

- a network of units obeying this decision rule will eventually reach "thermal equilibrium" and the relative probability of two global states will follow the **Boltzmann distribution**:

$$\frac{P_\alpha}{P_\beta} = e^{-(E_\alpha - E_\beta)/T}$$

  where P_α is the probability of being in the α^{th} global state and E_α is the energy of that state

- at low temperatures there is a strong bias in favour of states with low energy, but the time to reach equilibrium may be long.
- at higher temperatures the bias is not so favourable but equilibrium is reached faster.

What's so special about Boltzmann distributions?

1. they work well for doing constraint satisfaction searches.
2. they can be implemented very naturally in a parallel network.
3. they allow powerful general purpose learning rules (e.g. maximum likelihood estimation).
4. they are exactly what is required for "direct" Bayesian models in complex networks (i.e. models in which units stand for hypotheses and the probability of finding the unit on is the probability that the hypothesis is correct).

Simulated Annealing

- a good way to beat this trade-off is to start at a high temperature and gradually reduce it.
- this corresponds to annealing a physical system.
- this technique has proven useful when trying to satisfy multiple weak constraints.
- it will fail in cases where the best solution corresponds to a minimum that is deep, narrow and isolated.

The Learning Algorithm

- the Boltzmann Machine formulation leads to a domain-independent learning algorithm that modifies the connection strengths between units in such a way that the whole network develops an internal model which captures the underlying structure of the environment.
- the algorithm used for changing a weight depends on collecting statistics about the behaviour of the two units that the weight connects.
- to be capable of interesting computations, a network must contain non-linear elements that are not directly constrained by the input, and when such a network does the wrong thing it appears to be impossible to decide which of the many connection strengths is at fault – this is known as the **credit-assignment** problem.
- this problem can be solved within the BM formulation.
- by using the right stochastic decision rule and by running the net until "thermal equilibrium" at some finite temperature, a mathematically simple relationship between the probability of a global state and its energy is achieved.
- if the environment directly specifies the required probabilities P_α for each global state α, there is a straightforward way of converging on a set of weights that achieve those probabilities, provided such a set exists.
- this is not too interesting since it requires the probabilities of complete global states – the internal representation has already been decided by the environment.

- the interesting problem is to model the underlying structure of an environment.

Modelling the Environment

- units are divided into two functional groups

  1. a non-empty set of visible units,
  2. a possibly empty set of hidden units.

- **visible** units are the interface between the network and the environment (i.e. can be clamped).

- **hidden** units can never be clamped. They can be used to explain underlying constraints which cannot be represented by pairwise constraints.

- the structure of an environment can then be specified by giving the probability distribution over all 2^v states of the v visible units.

- it will be impossible to achieve a **perfect** model even if the network is totally connected.

- however, if the network uses the hidden units to capture regularities in the environment, it may achieve a good match to environmental probabilities.

- an information-theoretic measure of the discrepancy between the network's internal model and the environment is

$$G = \sum_{\alpha} P(V_\alpha) ln \frac{P(V_\alpha)}{P'(V_\alpha)}$$

  where $P(V_\alpha)$ is the probability of the α^{th} state of the visible units as determined by the environment and $P'(V_\alpha)$ is the corresponding probability when the network is running freely with no input.

- the **G** metric is called the information gain and is **zero** if and only if the distributions are identical, otherwise it is positive.

- to minimize G, it is therefore sufficient to observe p_{ij} and p'_{ij} when the network is at thermal equilibrium and to change the weight by an amount proportional to the difference between these two probabilities:

$$\Delta w_{ij} = \epsilon(p_{ij} p'_{ij})$$

  where ϵ scales the size of each weight change.

- this rule uses only locally available information.

- once G has been miminized the network will have captured the regularities in the environment, and these regularities will be enforced when performing completion.

Controlling the Learning

- there are a number of free parameters and possible variations in the learning algorithm:

  1. the size of ϵ which determines the size of each step taken for gradient descent
  2. the lengths of time over which p_{ij} and p'_{ij} are estimated

  which have a significant impact on the learning process.

The BM Learning Procedure

- $Phase^+$
  - For the entire set of I/O pairs:
    1. clamp both the input vector and the output vector.
    2. let the hidden units reach thermal equilibrium (at $T = 1$).
    3. once the system is close to equilibrium, keep it running for a few more cycles, during which a record is kept of how often two connected units are on at the same time (for all connections).
    4. p^+ is the fraction of time during the positive phase in which the two connected units are on at the same time at thermal equilibrium.
- $Phase^-$
  - For the entire set of I/O pairs:
    1. clamp the input units, but leave the hidden and output units free.
    2. let the network reach thermal equilibrium.
    3. once the system is close to equilibrium, keep it running for a few more cycles, during which a record is kept of how often two connected units are on at the same time (for all connections).
    4. p^- is the fraction of time during the negative phase in which the two connected units are on at the same time at thermal equilibrium.
- if the network produces a distribution in $phase^-$ that matches the distribution of output vectors clamped on it in $phase^+$, then the network is producing all the right answers.
- if p^+ and p^- differ for a given connection, the two probability distributions can be made to match better by changing that connection's weight.
- steepest descent is performed in this difference measure by changing the weight by an amount proportional to $p^+ - p^-$.

A Problem

- if the environment specifies only a small subset of possible patterns, then the only way to guarantee that certain configurations never occur is to give them infinitely high energy (i.e. infinitely high weights).
- the solution is to occasionally provide **noisy** input vectors.
- if the noise is small, the correct vectors will dominate the statistics, but every vector will have some chance of occurring and so infinite energies will not be needed.

Another Problem

- there is nothing in the learning algorithm to prevent it from creating an energy landscape that contains large energy barriers which prevent the network from reaching equilibrium.
- if this happens, the network may perform badly and the statistics that are collected will not be equilibrium statistics so there is no guarantee that the changes in the weights will improve G.
- to keep all the weights small, redefine the quantity to be minimized as:

$$G + h \sum_{ij} (w_{ij})^2$$

  where h is a coefficient that determines the relative importance of minimizing G and keeping the weights small.
- the effect of the extra term is to make all the weights decay towards zero by an amount proportional to their current magnitude – this ensures that large weights which are not important for achieving a low value of G tend to shrink.

The Speed of Learning

- even in a truly parallel machine, the learning would be slow because the gradient descent requires a great many annealings with different input vectors.
- this raises several issues:

  1. how does the learning time scale with the size of the problem?
  2. can the learning algorithm be generalized to exhibit "one-shot" learning?
  3. how much faster is the learning when the network is approximately correct for the task?
  4. do good solutions generally have a particular statistical structure?

How the Learning Time Scales

- the problem of scale is very complex because many factors have to be scaled at the same time and it is not obvious how they should be scaled.
- factors include:
  1. the ratio of hidden to visible units.
  2. the number of connections per unit.
  3. the number of constraints in which each visible unit is involved.
  4. the order of the underlying constraints.
  5. the compatibility of the constraints.

The XOR Example

- suppose an **environment** consisted of the following equiprobable vectors:

$$\{< 0,0,0 >, < 0,1,1 >, < 1,0,1 >, < 1,1,0 >\}$$

- rules that characterize the set:
  1. the third element is the exclusive-OR of the first two.
  2. the second element is the exclusive-OR of the first and third.
  3. the set of all triples of bits with even parity.
- the state of any one element, by itself, provides no information about whether another element should be one or zero.
- a single-level decision unit is unable to capture the distinquishing feature of all characterizations of this solution set: the state of two of the units, taken together, determines the third.
- Perceptrons are unable to compute the exclusive-OR function.
- a BM with only three interconnected units is unable to represent this solution set.
- but, decompose the solution set into:

$$S_1 = \{< 0,0,0 >, < 0,1,1 >, < 1,0,1 >\}\ S_2 = \{< 1,1,0 >\}$$

- S_1 can be represented by linking the first and second units to the third with positive weights and providing the third unit with a negative bias so that in the absence of input from either of the other units, the third unit will tend to be off.
- the network will now correctly find the elements of S_1 but fails on S_2, preferring $< 1,1,1 >$ over the correct state.

- by adding a hidden unit, this preference can be overridden.
- the hidden unit is given positive weights to the first two units, a negative weight to the third unit and a bias low enough that both of the first two units must be on for the hidden unit to be likely to come on.
- the magnitudes of the weights are chosen so that the negative weight between the hidden unit and the third unit overrides the positive weights between the third unit and the first two.
- the XOR example is a very simple illustration of a very hard general problem that a learning system must face: the task of discovering and representing the higher-order regularities of the solution set.
- hidden units are not directly constrained by the instances in a solution set and thus are available for reducing these higher-order constraints.

Distributed Representations

- an entity is represented by a pattern of activity distributed over many computing elements and each computing element is involved in representing many different entities.
- the knowledge is diffuse – this is good for fault tolerance, but it appears to make the design of modules to perform specific functions much harder.
- in a Boltzmann Machine, it corresponds to an energy minimum – the problem of creating a good collection of distributed representations is equivalent to the problem of creating a good "energy landscape".
- the Boltzmann Machine is unable to produce sequential behaviour - it can only respond to environmental changes (due to symmetry).
- a system could be composed of a number of internally symmetric modules that are asymmetrically connected to one another.
- each module could perform constraint-satisfaction searches, with the asymmetric inputs acting as boundary conditions that determine the particular problem it has to solve at any moment.
- it might be appropriate to use very different kinds of description at different time-scales.
- in the short term, the modules would perform parallel iterative constraint-satisfaction searches.
- in the longer term, the result of each search could be viewed as a single step in a strictly serial process, with each search setting up the boundary conditions for the next.
- this corresponds to the idea of a production system architecture in which all the heavy computational work is done by a parallel **recognition process** that decides which rule best fits the current state of working memory.

The Boltzmann Machine as a Neural Model

- one of the main reasons for studying Boltzmann Machines is that they bear some resemblances to the human **brain**.
- neurons are complex biochemical entities and binary units are not meant to be literal models but changing state asynchronously and using a probabilistic decision rule is not that bad a model of neural activity.
- the energy gap for a binary unit has a role similar to the membrane potential of a neuron.
- in Boltzmann Machines all connections are symmetrical - unlikely to be strictly true of neurons, but random asymmetry many act like Gaussian noise - so the model may still be valid.
- many problems in vision, speech recognition, associative recall and motor control can be formulated as searches.
- similiarity between different areas of cerebral cortex suggests that the same kind of massively parallel searches may be performed in many different cortical areas.

Boltzmann Machines and Bayesian Inference

- Bayes rule for updating the odds on hypothesis h given evidence e can be written:

$$\frac{p(h|e)}{p(\bar{h}|e)} = \frac{p(h)}{p(\bar{h})} \cdot \frac{p(e|h)}{p(e|\bar{h})}$$

  where $\frac{p(h)}{p(\bar{h})}$ is the "prior odds" on h and $\frac{p(e|h)}{p(e|\bar{h})}$ is the "evidence" supporting h.

- this can be rewritten to resemble the update rule for a Boltzmann unit:

$$p(h|e) = \frac{1}{1 + e^{-[log(priorodds) + log(evidence)]}}$$

Representing Probabilities

- instead of using a real number for $p(h)$, Boltzmann machines use a binary variable that adopts the "true" value with probability $p(h)$.
- this is a "direct" representation of probability.

Standard Problems with Bayesian Inference

1. it is hard to assign a priori probabilities to every possible state of affairs. Probabilities must be consistent and easily expressed.
   - represent the probability distributions implicitly by potential functions.

$$P_\alpha = ke^{-E_\alpha}$$

2. it is hard to compute posterior probabilities given the prior distribution and the evidence.
    - use simulated annealing to get a good approximation. To save time, be content with probability matching.
    - approach equilibrium at a finite temperature, so that good solutions are much more likely than poor ones – much easier than finding the best solution.

Coping with non-independent sources of evidence

- each unit adds up the evidence coming in along each input line – so it assumes independence.
- if there is a set of hidden units, they can be used to make the independence assumption be a good approximation.
- first decide on the rule for combining evidence, then discover representations that make this rule work.

The Bell-core Boltzmann Chip

- Annealing
    - use analog circuitry with real noise which can be varied by a gain control to implement a very fast approximation to annealing. Have many "neurons" on a single VLSI chip.
    - each weight is a set of field effect transistors of size 1,2,4,8,16. Each transistor can be activated by a digital charge on the gate. The digital charges are stored in RAM cells.
    - the learning procedure uses digital counters to decide whether to increment or decrement a weight. But the "annealing" is analog so it's very fast.

How to Implement Weights on Chips

- Set of FETs
    - easy but takes a large area.
- Floating gates
    - the charge on a floating gate can last for years. It is a very compact method.
    - UV light is used to allow electrons to get onto the gate. Carver Mead has a very simple version of this working.
- Resistors
    - very compact for fixed weights, but no good for learning.

Reference Material

1. Hinton, Geoffrey E., Terrence J. Sejnowski and David H. Ackley, **Boltzmann Machines: Constraint Satisfaction Networks that Learn**, Technical Report CMU-CS-84-119, May 1984.

2. Simpson, Patrick K., **Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations**, 1990.

    - Chapter 5: ANS Paradigms and Their Applications and Implementations, pages 120-127.

3. Hinton, Geoffrey, **Lecture Notes for CS2535: Neural Networks**, University of Toronto, 1989.

4. Recce, Michael, **Connectionist Models: Background and Emergent Properties**.