CS 790R Seminar

Modeling & Simulation

Neural Networks 2 – Attractor Networks

Milind Zirpe

March 1, 2006

Department of Computer Science & Engineering
University of Nevada, Reno

Spring 2006

Discussion

- Flake (1998), Chapter 18
- Bar-Yam (1997), Sections 2.1 and 2.2
- Hopfield (1982)

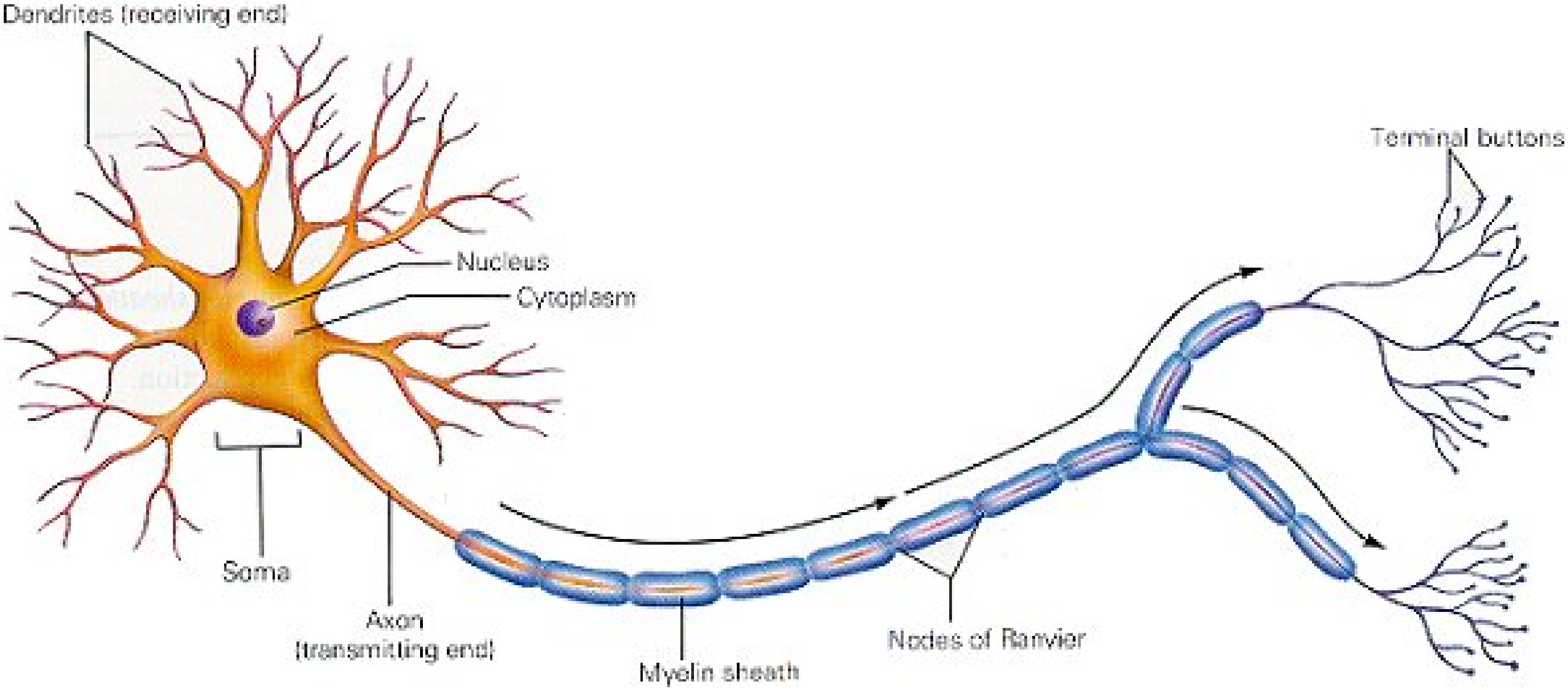
Introduction

- Biological neuron
- Associative memory
- Hebbian learning
- McCulloch-Pitts neuron
- Attractor networks (or Hopfield networks)

Biological neuron

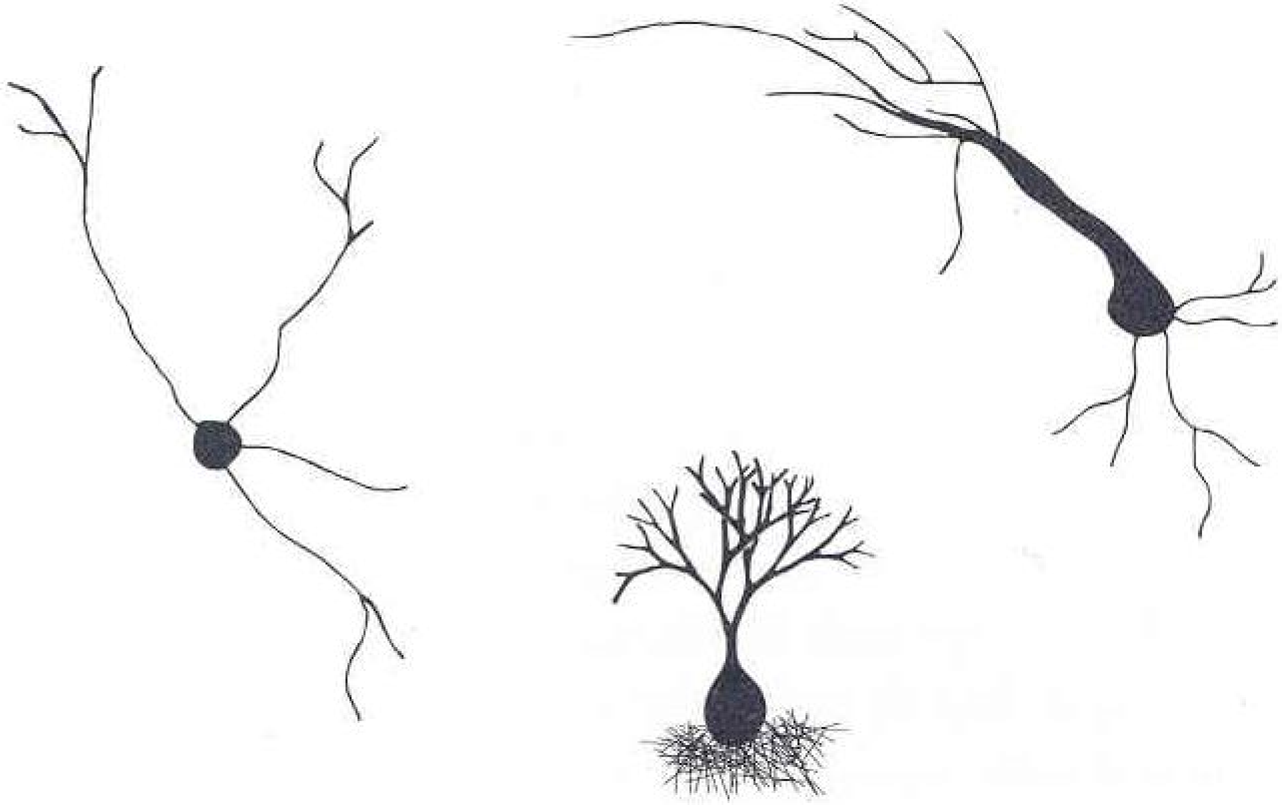
THE MAJOR STRUCTURES OF THE NEURON

The neuron receives nerve impulses through its dendrites. It then sends the nerve impulses through its axon to the terminal buttons where neurotransmitters are released to stimulate other neurons.



Biological neuron

Figure 2.1.1 Several different types of neurons adapted from illustrations obtained by various staining techniques.



Biological neuron

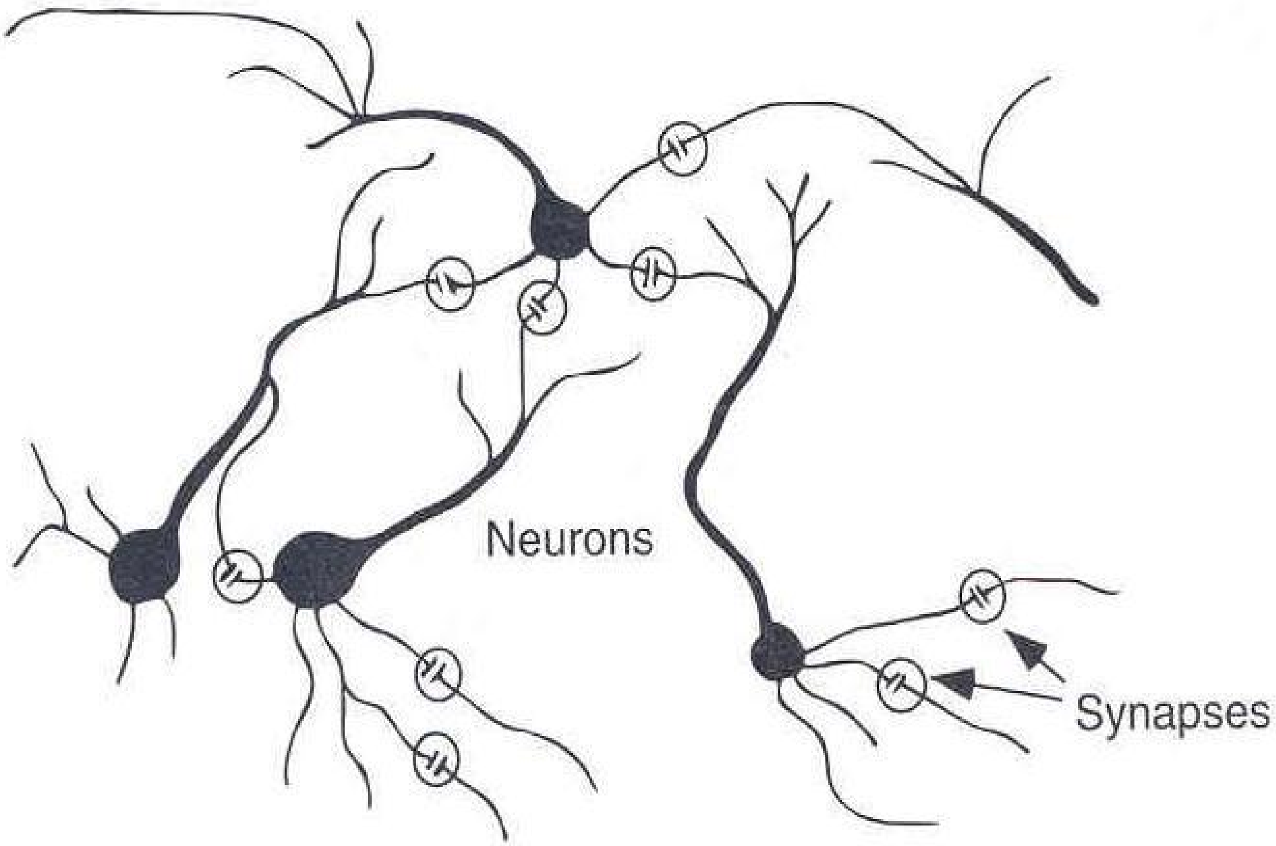


Figure 2.1.2 Schematic illustration of a biological neural network showing several nerve cells with branching axons. The axons end at synapses connecting to the dendrites of the next neuron that lead to its cell body. This schematic illustration is further simplified to obtain the artificial network models.

Associative memory

- Computers: Memory referenced by location.
- Humans: Memory referenced by Content. (e.g.: What *1960s rock band* with *four members* was named after an *insect* and started the “*British* Invasion” ?).
- This sort of Content-addressable memory is also referred to as Associative memory.

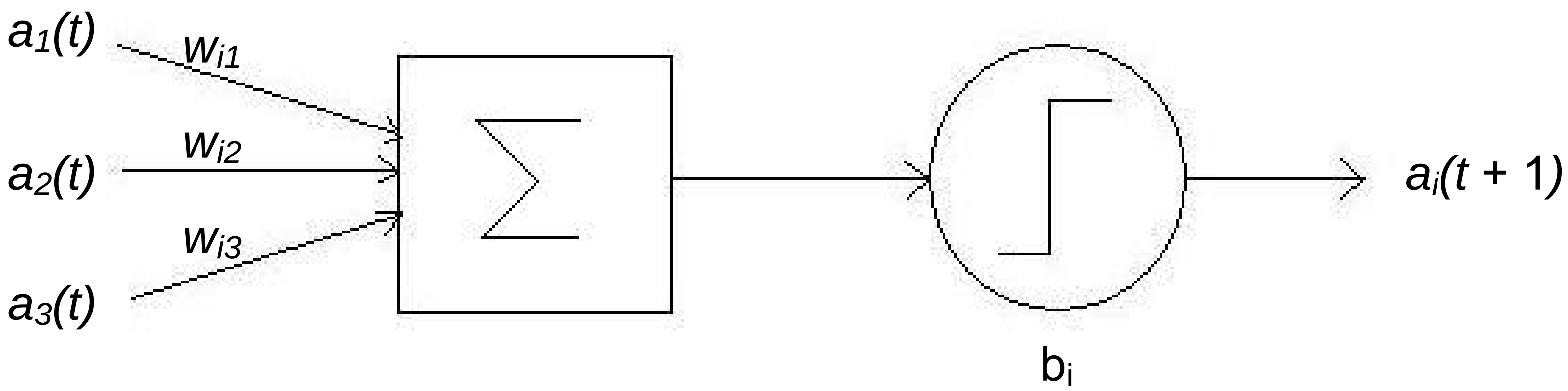
Hebbian learning

- Hebbian learning or Hebbian imprinting:

  When two connected neurons fire (or don’t fire) at a particular time, an excitatory synapse between them is strengthened and an inhibitory synapse is weakened.

  Conversely, vice versa happens if one of the neuron fires and other doesn’t.

McCulloch-Pitts neuron



- $a_i(t)$: Activation value of neuron i at time t.
- w_{ij}: Strength of synapse connecting neuron j to neuron i.
- b_i: Threshold that neuron i ’s net input must exceed in order to fire.
- $\Theta(x)$: Nonlinear unit step function: 1 (“*on*” state) if $x \geq 0$, 0 (“*off*” state) if $x < 0$.

McCulloch-Pitts neuron

- A neuron’s state or activation, $a_i(t)$ is a function of a weighted sum of all of the incoming signals to i^{th} neuron:

$$a_i(t+1) = \Theta(\sum_{j=1}^{n} w_{ij} \times a_j(t) - b_i)$$

where, Θ(x): Nonlinear unit step function:

1 (“*on*” state) if $x \geq 0$, 0 (“*off*” state) if $x < 0$.

McCulloch-Pitts neuron

- Modification to the previous update rule:

$$a_i(t+1) = \mathrm{sgn}(\sum_{j=1}^{n} w_{ij} \times a_j(t) - b_i)$$

where, sgn(*x*): Sign function:
1 (“*on*” state) if *x* ≥ 0, -1 (“*off*” state) if *x* < 0.

- Just a mathematical convenience for two reason:
  – Both (0,1) and (-1,1) representations are equivalent.
  – New representation is not very biologically plausible, since real neurons cannot inhibit other real neurons in this precise manner.

McCulloch-Pitts neuron

Q. How to update the activation rules?

- Synchronous:
  - Simultaneous calculation of next activation value.
  - Completely deterministic but unrealistic.
- Asynchronous:
  - Update neurons randomly.
  - More realistic.
  - Care has to be taken to avoid neglecting the updating of neurons.

McCulloch-Pitts neuron

Model: Recalling a pattern from many stored patterns.

- Hebbian learning:
  - If *i* and *j* both are either *on* or *off* at the same time, then w_{ij} should be positive.
  - If *i* and *j* have different activation values then w_{ij} should be a negative weight.
- Memory represented as a vector of variables, x_i, that have either -1 or 1 values.
- Number of neurons = Number of x_i terms.
- Weights: (between -1 to 1)

$$w_{ij} = \frac{1}{n}\sum_{k=1}^{p} x_i^k x_j^k$$, where x_i^k is i^{th} component of k^{th} pattern.

McCulloch-Pitts neuron

Model: cont…

- Assuming that all patterns that are stored, are drawn from a random sample.
- If we set *$a_j(t)$* terms equal to a stored pattern, say x_i^l, then next state of network should be equal to x_i^l terms.
- We need to ensure that *h_i* (net input for each neuron *i*) has same sign as x_i^l. We get *h_i* as:

$$h_i = x_i^l + \frac{1}{n}\sum_{j=1}^{n}\sum_{k\neq l}^{p} x_i^k x_j^k x_j^l$$

If terms inside summation are uncorrelated, then they will cancel each other out and we retrieve the pattern. Otherwise we can partially correct using *b_i* terms:

$$b_i = -\frac{1}{2n}\sum_j \sum_k x_i^k x_j^k$$

McCulloch-Pitts neuron

Example: Recalling letters.

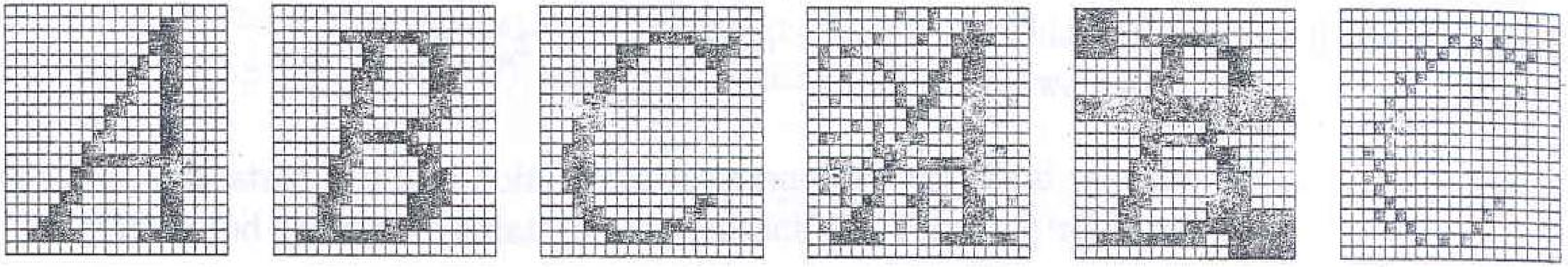


Figure 18.3 Bitmapped images of letters from the alphabet: The first three are clean version that are used as patterns to be stored. The last three are used as seed images that the associative memory must use to recall one of the first three.

- 20x20 grid of bits, 400 neurons, black = “*on*” or +1 and white = “*off*” or -1.
- Asynchronous update of neurons.

McCulloch-Pitts neuron

Example: Recalling letters.

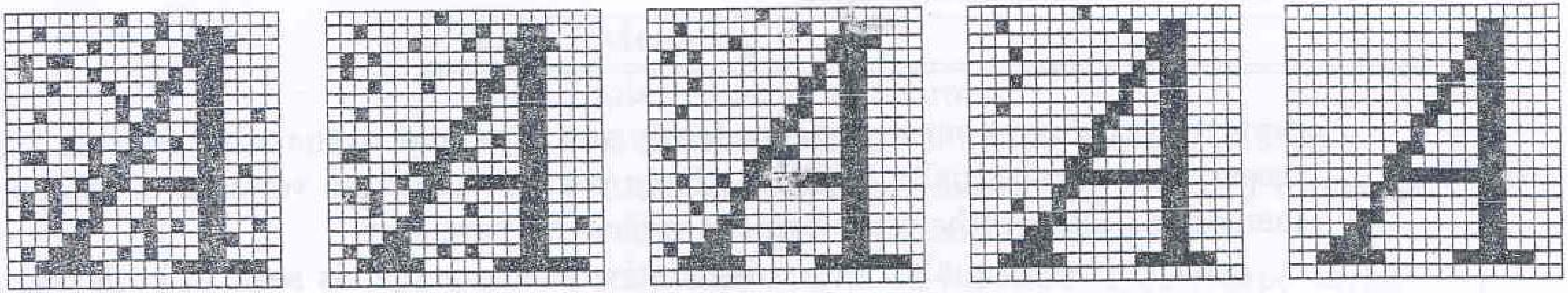


Figure 18.4 Recalling the letter A from a damaged seed image

McCulloch-Pitts neuron

Observations:

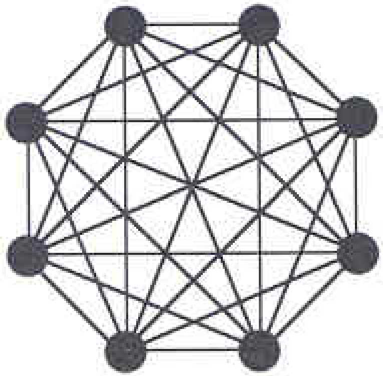
- Feedback neural networks: are artificial neural networks of this kind, which have a collection of neurons that can connect to any other neuron in the system.
- Has a discrete state and changes in discrete time steps.
- At some final time *T*, the system state will be such that, applying the update rules for any more time steps will result in same set of activations.
- The final converged state of a neural network can represent the answer to a question, performing a sort of analog computation.
- Number of weights is much larger than information in stored patterns. Correct this by removing weights smaller than a threshold, merging weights and removing redundancy of stored patterns.

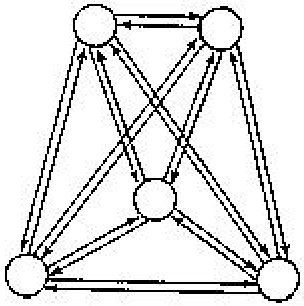
McCulloch-Pitts neuron

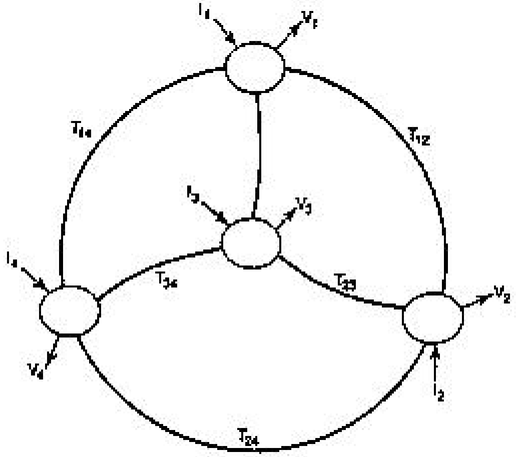
Observations:

- Whole system can be implemented as a collection of very simple parallel computers.
- Fault tolerant: Associative memories are not stored in any one place or weight. (65% of weights were destroyed without adversely affecting the network's performance).
- All associative memories are prone to recalling spurious memories that are a composite of many of the stored patterns.

Attractor or Hopfield networks







Simplified 4-neuron Hopfield network

Dots represent the neurons and lines represent the synapses.

- Synapses are symmetric carrying equal influence in both directions. $T_{ij} = T_{ji}$.
- No self-action by a neuron. $T_{ii} = 0$.
- Binary variables for neuron activity values $U_i(t+1) = \pm 1$.
- The artificial neurons have a continuous state (internal and external) and evolve continuously over time.

Attractor or Hopfield networks

- Change in internal state of neuron is:

$$\frac{dU_i}{dt} = \sum\nolimits_{j=1}^{n} T_{ij} V_j + I_i - \frac{U_i}{\tau}$$

- After approximating to discrete system:

$$U_i(t+1) = U_i(t) + \Delta t(\sum_{j=1}^{n} T_{ij} V_j(t) + I_j - \frac{U_i(t)}{\tau}$$

- U_i: Internal state of neuron *i*.
- V_i: External activation or visible state of neuron *i*.
- T_{ij}: Strength of synapse connecting neuron *j* to neuron *i*.
- I_i: External input injected into neuron *i*.
- g(x): Sigmoidal activation function: 1/(1+exp(-x))
- τ: Inverse decay term for internal state.
- Δt : is the simulation time-step increment.

Attractor or Hopfield networks

- Activation function of neuron is g(x), known as sigmoid function (S-shaped).

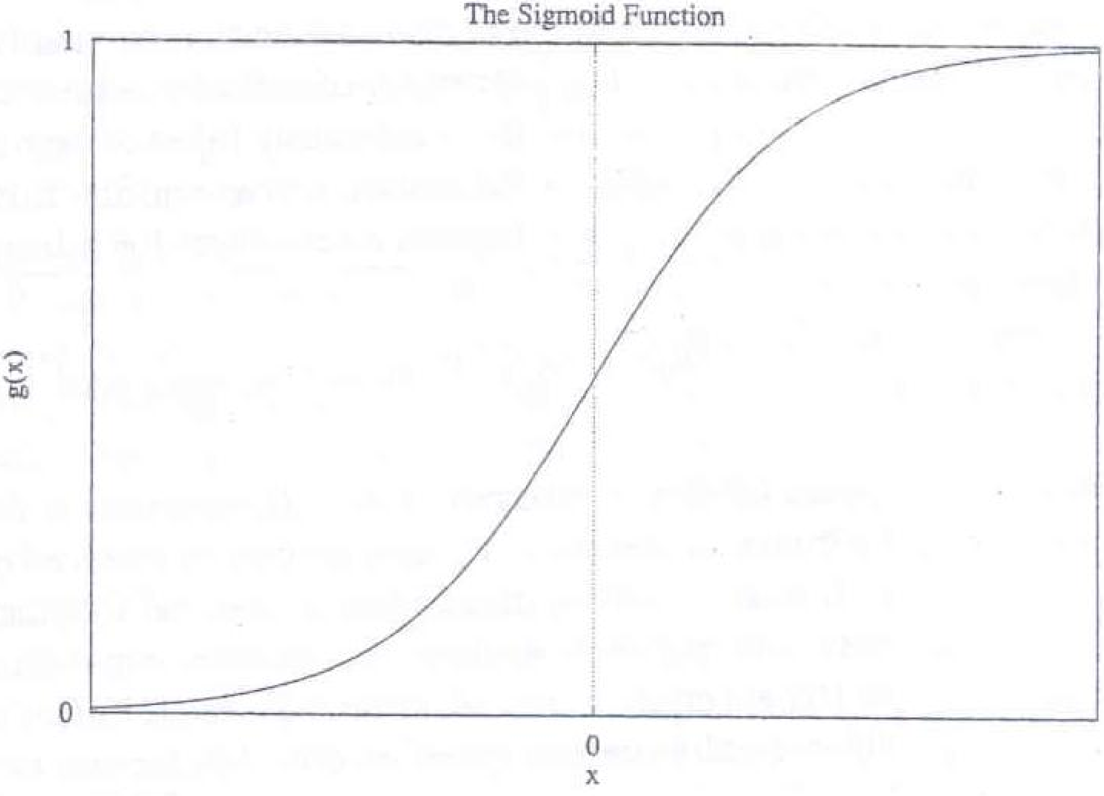


Figure 18.5 The sigmoid activation function, $g(x)$

$$V_i(t) = g(U_i(t))$$

Attractor or Hopfield networks

- Operation:
  - A pattern of neural activities (input) is given to the network.
  - Network is evolved by updating neurons several times until a steady state (local energy minimum) or pre-specified number of updates is reached.
  - Then the state of network is read as output.
  - The next pattern is then imposed on the network and same as above.
- Training (Hebbian Imprinting):
  - Synapse is changed in direction of excitatory if both neurons were either "*on*" or "*off*".
  - Synapse is changed in direction of inhibitory if one of the neurons is "*on*" and other is "*off*".
  - Training consists of imprinting a set of "*p*" selected neuron firing patterns.

Attractor or Hopfield networks

- Energy analog:

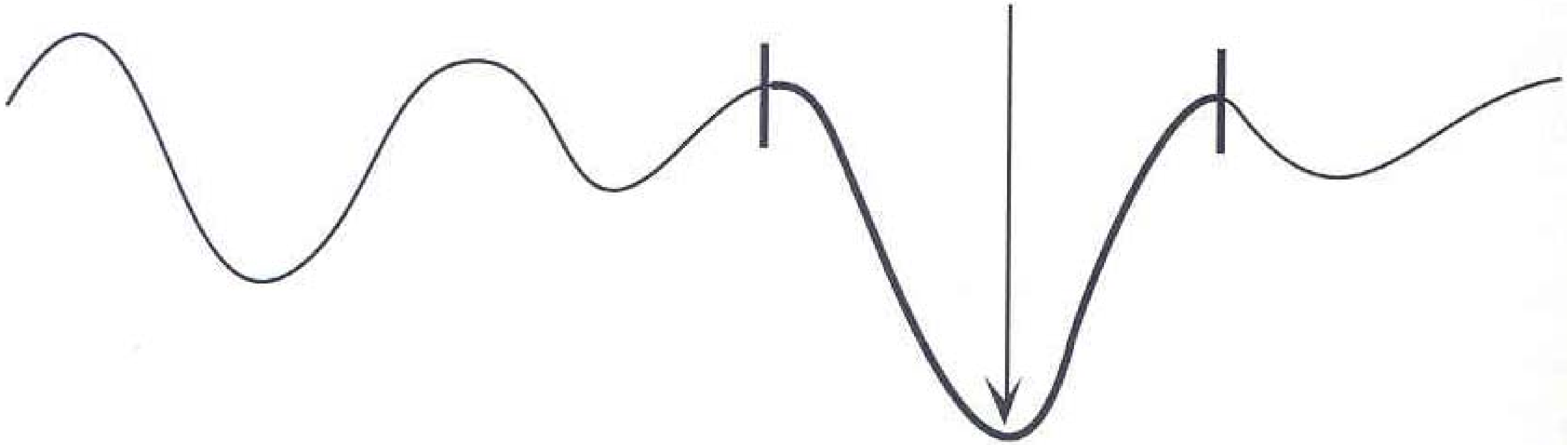


Figure 2.2.1 Schematic illustration of the energy analog of imprinting on an attractor network. Imprinting a pattern lowers its energy and the energy of all patterns in its vicinity. This creates a basin of attraction. If we initialize the network to any pattern within the basin of attraction, the network will relax to the imprinted pattern by its own neural evolution. The network acts as a memory that is "content-addressable." When a pattern is imprinted we can recover it by starting from partial information about it (see Fig. 2.2.2). This is also a form of associative memory. ∎

Attractor or Hopfield networks

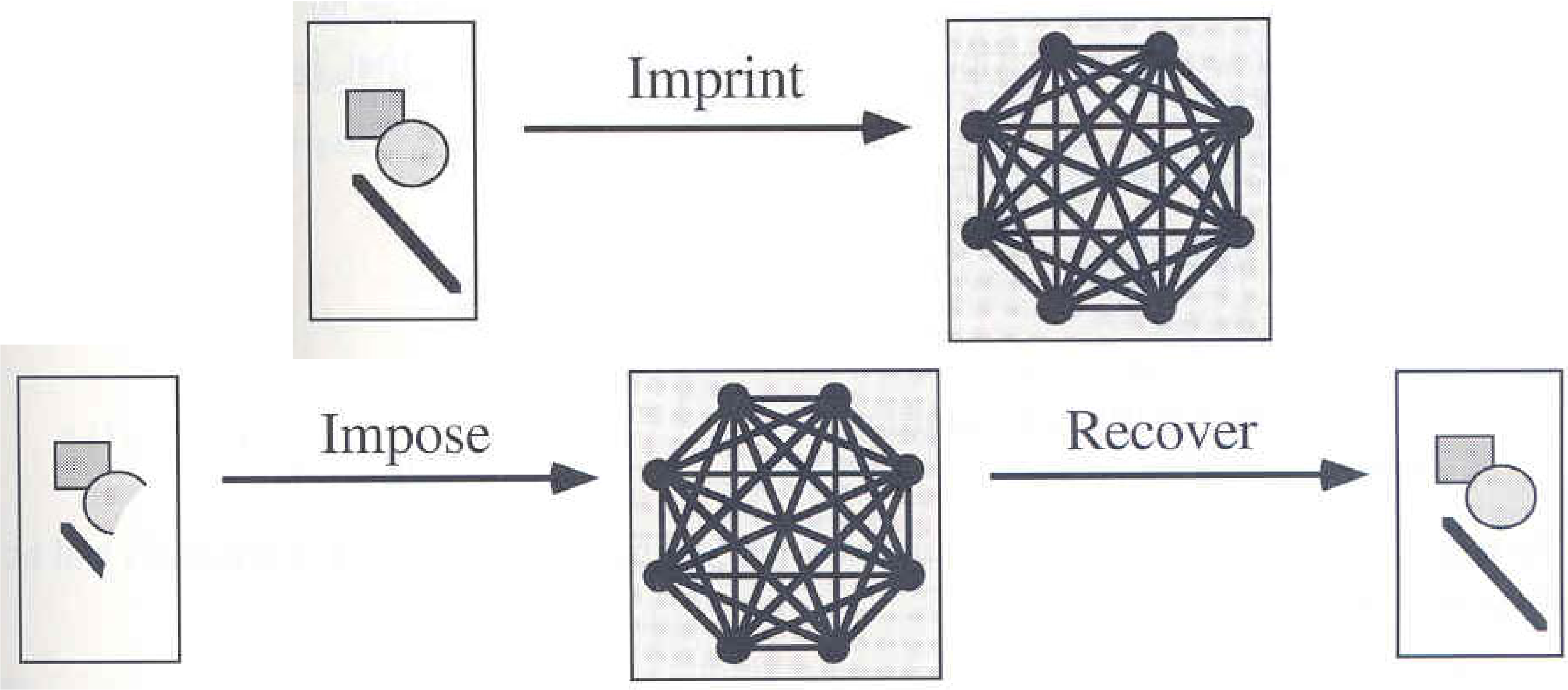


Figure 2.2.2 Schematic operation of the attractor network as a content-addressable memory. Imprinting a pattern on the network in the training stage (top) enables us to use the network as a content-addressable memory (bottom). By imposing a pattern that has a significant overlap with the imprinted pattern the original pattern can be recovered. This is analogous to being able to complete the sentence "To be or not to be ..."

Attractor or Hopfield networks

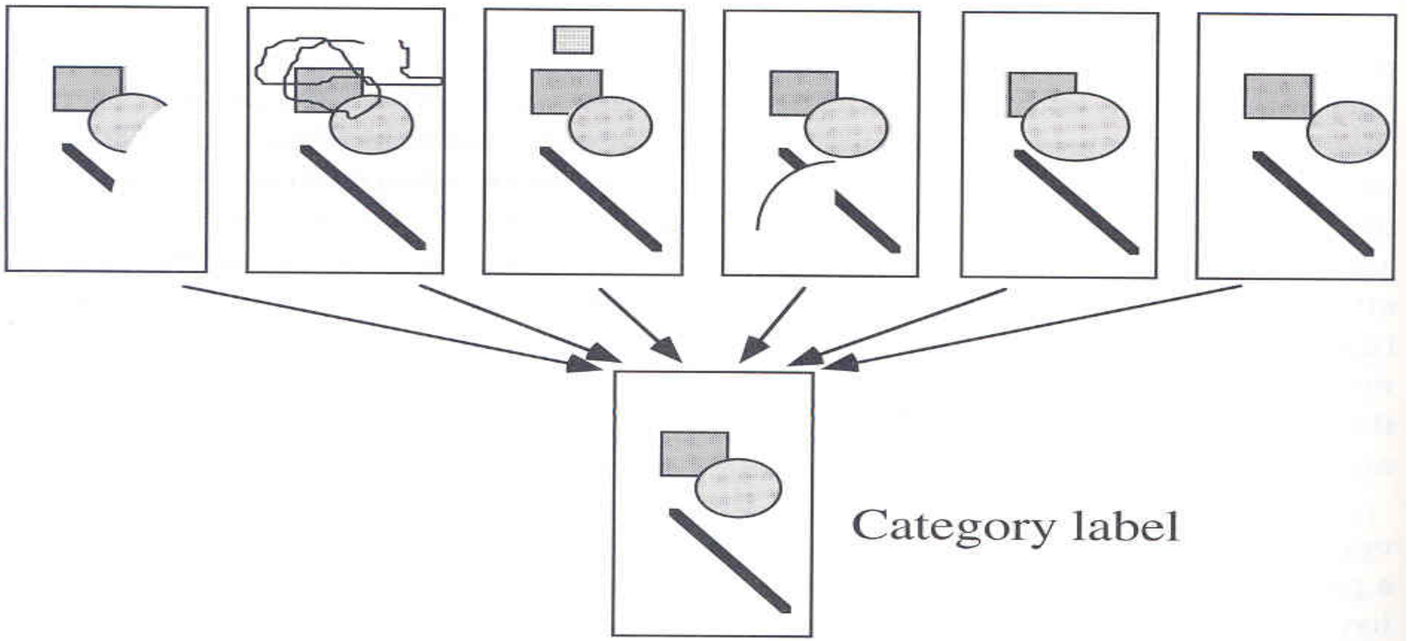
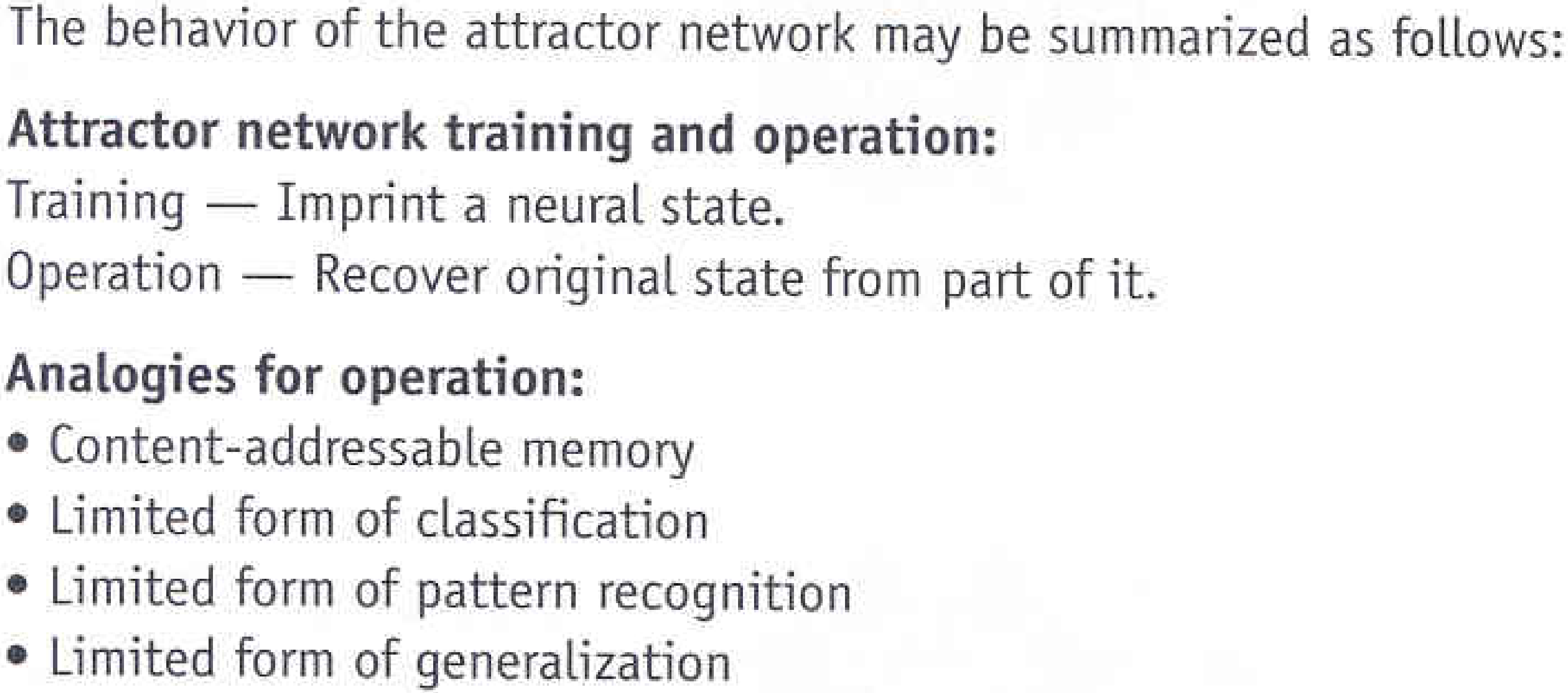


Figure 2.2.3 The neural dynamics of an attractor network maps a variety of patterns onto an imprinted pattern. This is equivalent to a classification of patterns by a category label, which is the imprinted pattern. The category of patterns labeled by the imprinted pattern is its basin of attraction. Classification is also a form of pattern recognition. Moreover, we can say that the basin of attraction is a generalization of the imprinted pattern. Thus the attractor network has properties very unlike those of a conventional computer memory, which is accessed using a numerical address that is distinct from the memory itself. It works much more like human memories that are accessed through information related to the information that is sought after.

Attractor or Hopfield networks



The behavior of the attractor network may be summarized as follows:

Attractor network training and operation:

Training — Imprint a neural state.

Operation — Recover original state from part of it.

Analogies for operation:

- Content-addressable memory
- Limited form of classification
- Limited form of pattern recognition
- Limited form of generalization

Attractor or Hopfield networks

Observations:

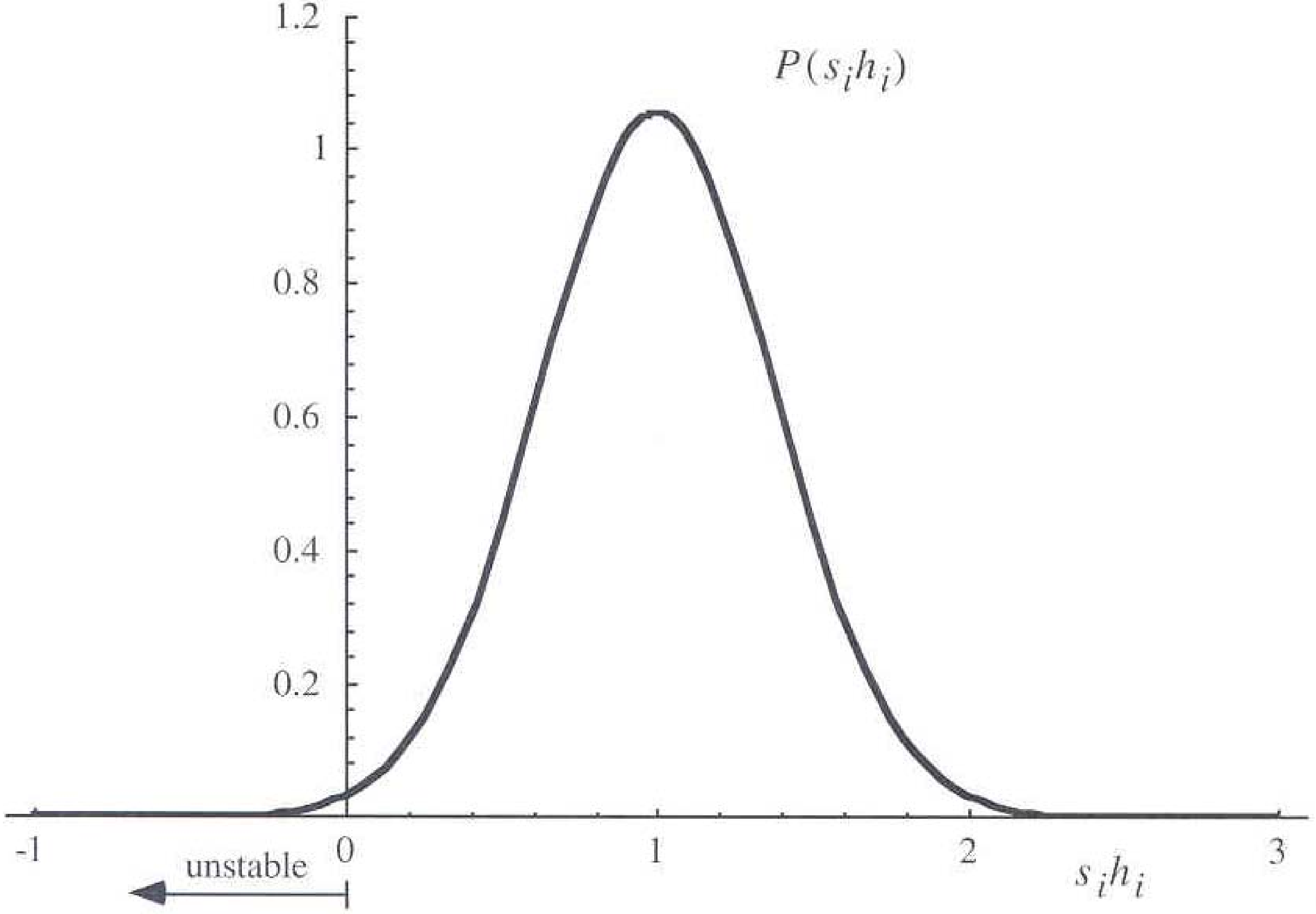
- Single imprinted pattern:
  - The pattern and its inverse is automatically stored. (Hebbian learning).
  - Imprinted pattern is the stable or fixed point of network dynamics.
  - Even if initial pattern is non-correlated, it leads to the stored imprinted pattern. Since sum over N uncorrelated ±1 neuron values is $\sqrt{N}$, which places pattern within the “basin of attraction” (either for imprinted pattern or its inverse).
  - “Basin of attraction” is large.

Attractor or Hopfield networks

- Couple of imprinted patterns:
  - Size of “basin of attraction” is equal to the Hamming distance d(s,s’) between two patterns as the number of neurons that differ between them.
  - Retrieval depends on proximity of initial state with pattern that will be retrieved and the number of neurons in the network.

Attractor or Hopfield networks

- Signal-to-noise analysis: Note: $s_i = U_i$ and h_i post-synaptic potential.



This figure illustrates the signal-to-noise analysis of stability of an imprinted pattern.

Attractor or Hopfield networks

- Signal-to-noise analysis:

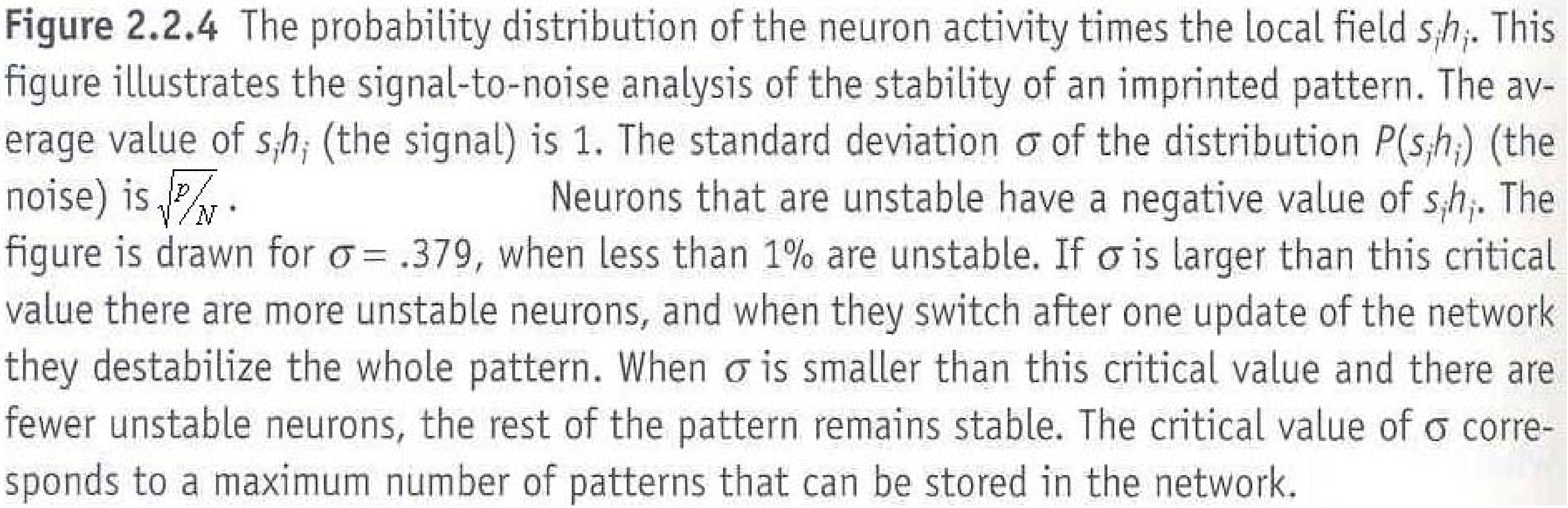


Figure 2.2.4 The probability distribution of the neuron activity times the local field s_ih_i. This figure illustrates the signal-to-noise analysis of the stability of an imprinted pattern. The average value of s_ih_i (the signal) is 1. The standard deviation σ of the distribution $P(s_ih_i)$ (the noise) is $\sqrt{p/N}$. Neurons that are unstable have a negative value of s_ih_i. The figure is drawn for $\sigma = .379$, when less than 1% are unstable. If σ is larger than this critical value there are more unstable neurons, and when they switch after one update of the network they destabilize the whole pattern. When σ is smaller than this critical value and there are fewer unstable neurons, the rest of the pattern remains stable. The critical value of σ corresponds to a maximum number of patterns that can be stored in the network.

Attractor or Hopfield networks

- Simulations: % Stability of imprints.

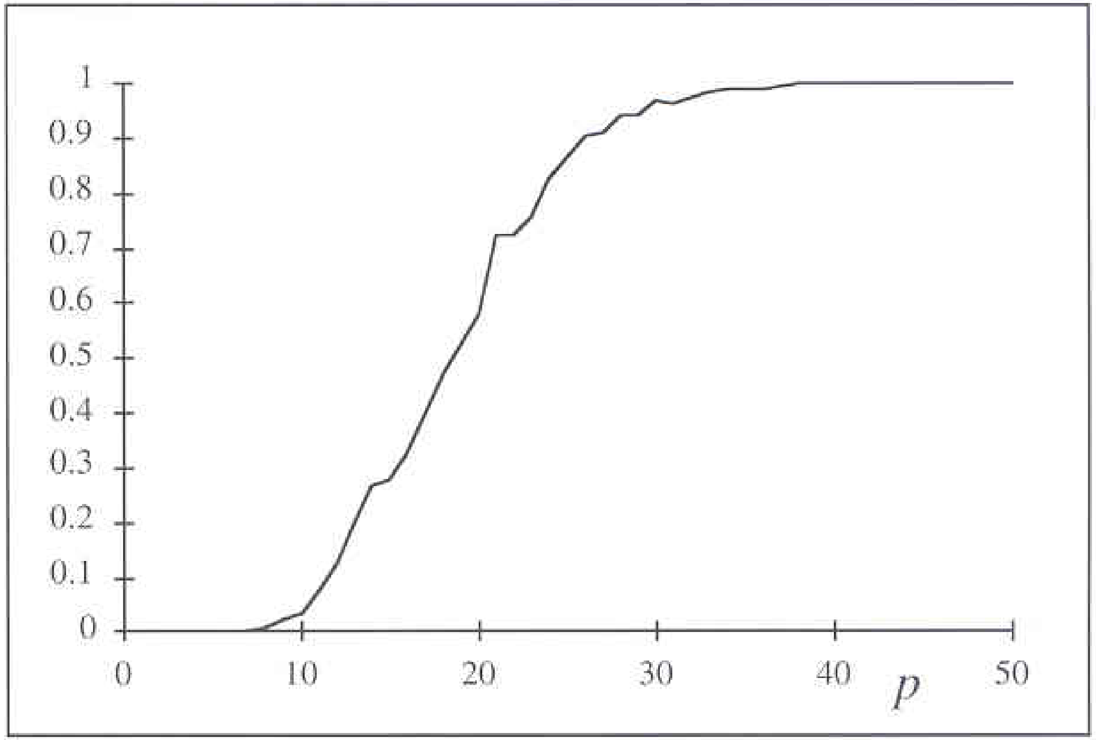


Figure 2.2.5 Fraction of unstable imprints as a function of the number of imprints p on a neural network of 100 neurons using Hebbian imprinting. For p less than 10 the stability of all of the stored patterns is perfect. Above this value the percentage of unstable patterns increases until all patterns are unstable.

Attractor or Hopfield networks

- Simulations:

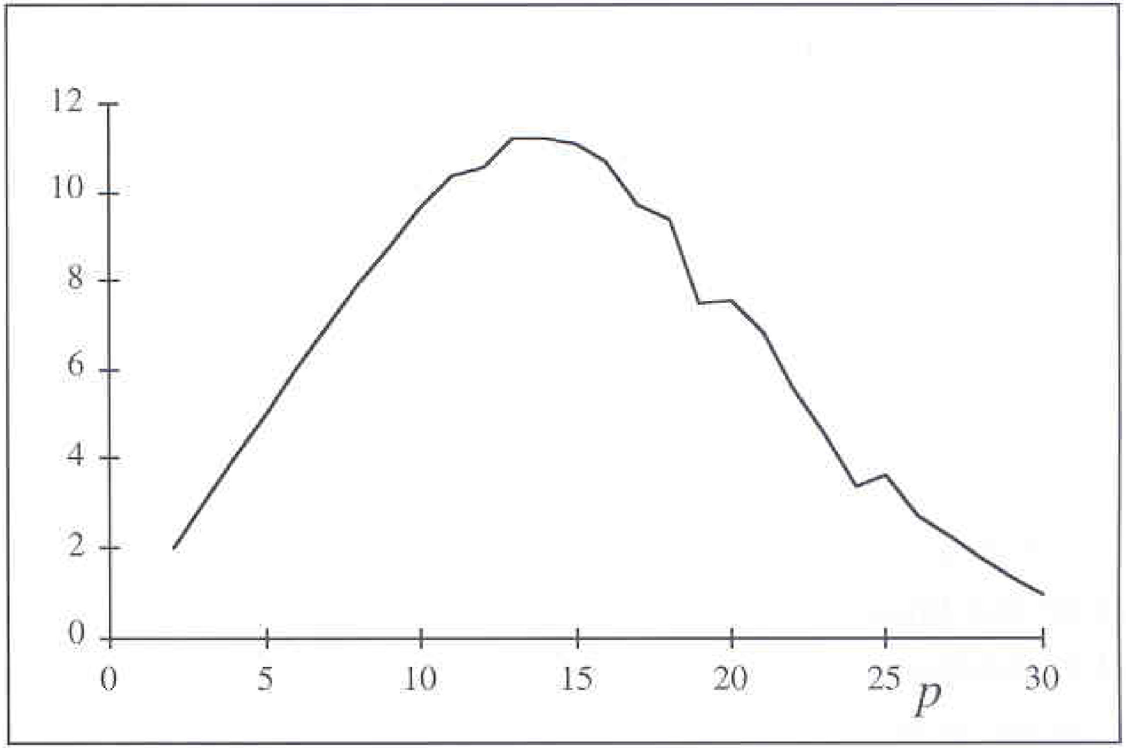


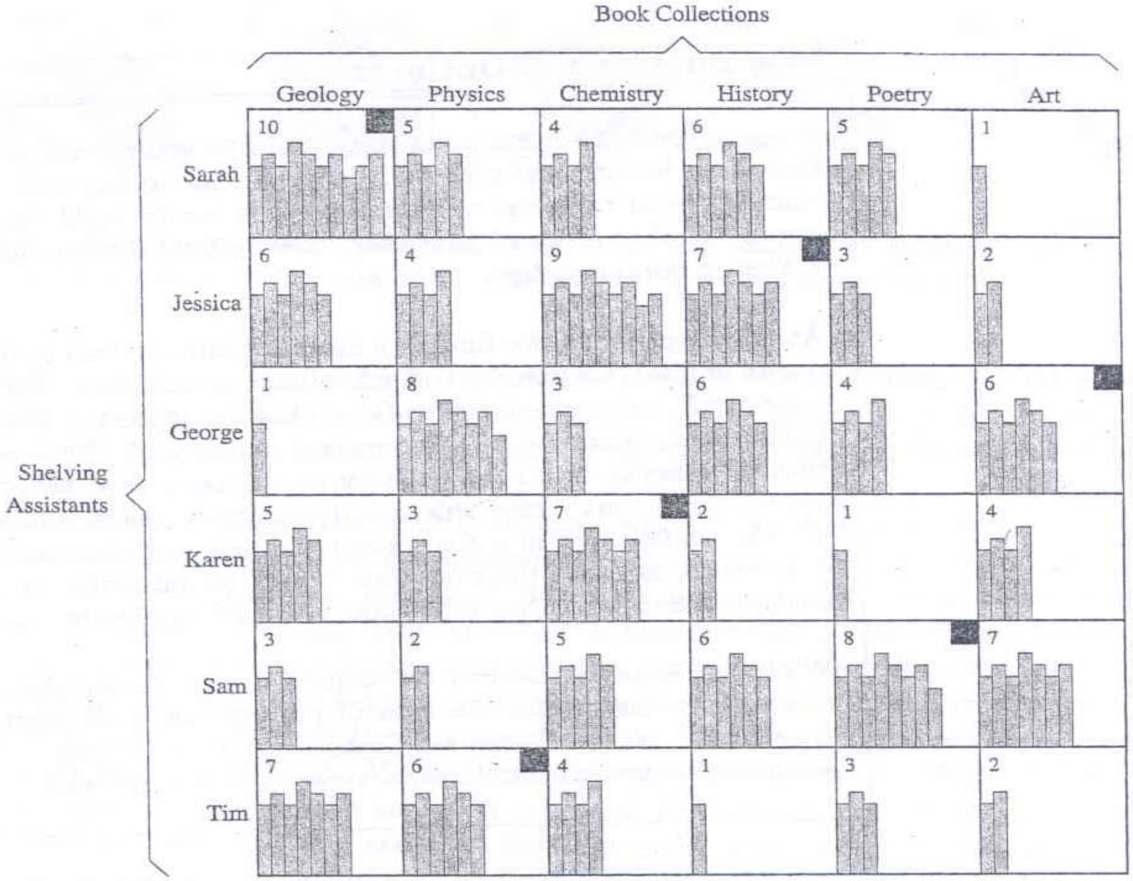
Figure 2.2.6 Number of stable imprints as a function of the number of imprints *p* on a neural network of 100 neurons using Hebbian imprinting. For *p* less than 10 all patterns are stable. The maximum number of stable imprinted patterns is less than 12. Above 15 imprints the number of stable patterns decreases gradually to zero. However, throughout this regime the basins of attraction of the patterns are very small and the system is not usable as a memory.

Attractor or Hopfield networks

- Overload and spurious states:
  - For low storage, p<<N, the neurons have a signal much greater than noise. So pattern will be stable.
  - Basin of attraction of the spurious patterns is shallower and smaller than that of imprinted patterns.
  - Ambiguities are solved on statistical basis.

Attractor or Hopfield networks

- Example: Task assignment problem.



	Geology	Physics	Chemistry	History	Poetry	Art
Sarah	10 ■	5	4	6	5	1
Jessica	6	4	9	7 ■	3	2
George	1	8	3	6	4	6 ■
Karen	5	3	7 ■	2	1	4
Sam	3	2	5	6	8 ■	7
Tim	7	6 ■	4	1	3	2

Figure 18.6 The task assignment problem: Black squares in the entries denotes the optimal assignment with a total shelving rate of 44.

Attractor or Hopfield networks

Example: Task assignment problem.

- K-out-of-n rule:
  - For a converging configuration: k=number of “on” neurons.
  - Weight = -2, if $i \neq j$

    0, if $i = j$

    (2k-1), otherwise
  - External input for each neuron centered on 2(2k-1). (Each neuron is in two clusters, one row and one column).
  - Overall average of all external inputs is 2.
- Random initial state close to 0.5.

Attractor or Hopfield networks

- Example: Task assignment problem. (k-out-of-n rule)

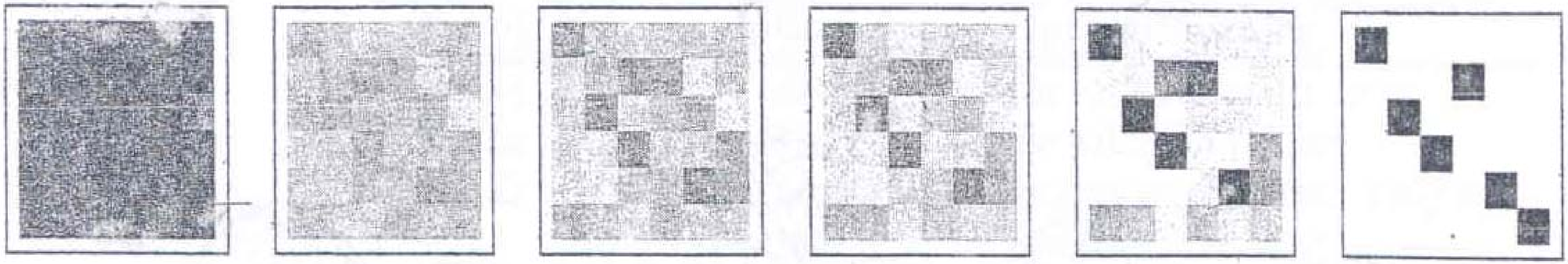


Figure 18.7 Computing a neural solution to the task assignment problem: This particular solution yields a total rate of 42, which is just less than the optimal solution.

- Lighter color are more closer to 0 and darker colors are more closer to 1.

Conclusion

- A global property such as collection of neural activations that compose a distributed memory may emerge from only local interactions.
- Should not expect the Hopfield network to recognize patterns more than 15% of total neurons in the network.
- We can determine storage capacity by measuring the stability of patterns that are imprinted on the network.
- Robust, as it does not depend strongly on precise details of the model (damage to few neurons doesn’t hamper the network severely).
- Complex computational attributes modeling from the collective behavior of large number of simple processing elements (neurons).
- Hopfield networks are used in associative or content-addressable memory, pattern recognition and combinatorial cost-optimization problems.

Thank You