UNIT 5

Neuro Dynamics:

The application of dynamical systems theory to neuro-dynamics provides a powerful framework for understanding the complex temporal behavior of neural systems

Neurodynamic: The Brain as a Dynamical System

- Neurodynamic applies dynamical systems theory to the study of neural activity.
- The brain, at various levels (from individual neurons to large-scale networks), exhibits complex temporal patterns.
- Key aspects:
  - **Neuronal dynamics:** Individual neurons' membrane potentials and firing rates change over time, and these changes can be modeled using differential equations (e.g., the Hodgkin-Huxley model).
  - **Network dynamics:** The interactions between neurons create complex patterns of activity, such as oscillations and synchronized firing.
  - **Cognitive functions:** Many cognitive processes, such as memory, perception, and decision-making, can be understood in terms of the dynamics of neural networks.

Key Applications in Neuro-dynamics

- **Modeling neuronal activity:** Dynamical systems models are used to simulate the behavior of individual neurons and neural circuits.
- **Understanding brain rhythms:** Brain oscillations (e.g., alpha, beta, gamma rhythms) can be analyzed using dynamical systems techniques.
- **Analyzing neural networks:** Dynamical systems theory provides tools for studying the stability, synchronization, and other properties of neural networks.
- **Investigating cognitive processes:** Dynamical systems models are used to explore how neural dynamics underlie cognitive functions.
- **Understanding neurological disorders:** Changes in neural dynamics are implicated in various neurological disorders, and dynamical systems theory can help to understand these changes.

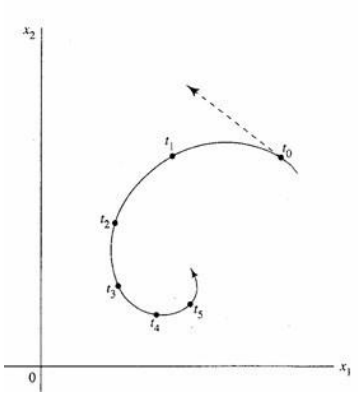


FIGURE 14.1 A two-dimensional trajectory (orbit) of a dynamical system.

Stability of Equilibrium States:

Stability:

- The stability of these equilibrium states determines how the neural system responds to perturbations.
- A stable equilibrium means the system will return to that pattern after a small disturbance.
- An unstable equilibrium means a small disturbance will cause the system to move to a different state.

Equations and Analysis:

- **Differential Equations:**
  - The dynamics of neural systems are often described by differential equations. For example:
    - **Hodgkin-Huxley equations:** Model the membrane potential of a neuron.
    - **Rate equations:** Describe the average firing rates of neurons in a network.
  - These equations define how the state of the system changes over time.

Linearization and Eigenvalues:

- To analyze the stability of an equilibrium point, we often linearize the differential equations around that point.
- The eigenvalues of the resulting Jacobian matrix determine the stability:
  - If all eigenvalues have negative real parts, the equilibrium is stable.
  - If any eigenvalue has a positive real part, the equilibrium is unstable

DEFINITION 1. The equilibrium state $\bar{\mathbf{x}}$ is said to be uniformly stable if for any given positive ϵ there exists a positive δ such that the condition

$$\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$$

implies

DEFINITION 2. The equilibrium state $\bar{\mathbf{x}}$ is said to be convergent if there exists a positive δ such that the condition

$$\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$$

implies that

$$\mathbf{x}(t) \to \bar{\mathbf{x}} \quad \text{as } t \to \infty$$

DEFINITION 3. The equilibrium state $\bar{\mathbf{x}}$ is said to be asymptotically stable if it is both stable and convergent.

Here we note that stability and convergence are independent properties. It is only when both properties are satisfied that we have asymptotic stability.

DEFINITION 4. The equilibrium state $\bar{\mathbf{x}}$ is said to be asymptotically stable or globally asymptotically stable if it is stable and all trajectories of the system converge to $\bar{\mathbf{x}}$ as time t approaches infinity.

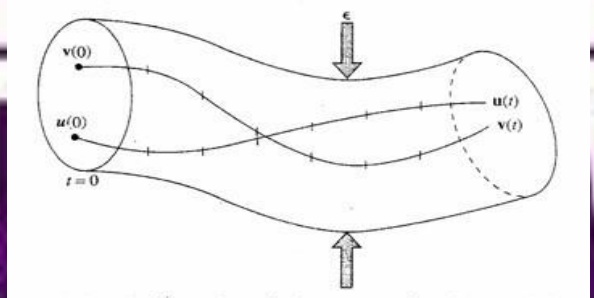


FIGURE 14.5 Illustration of the notion of uniform stability

Dissipative systems are generally characterized by the presence of attracting sets or manifolds of dimensionality lower than that of the state space. By a "manifold" we mean a k-dimensional surface embedded in the N-dimensional state space, which is defined by a set of equations:

$$M_j(x_1, x_2, \ldots, x_N) = 0, \quad \begin{cases} j = 1, 2, \ldots, k \\ k < N \end{cases}$$

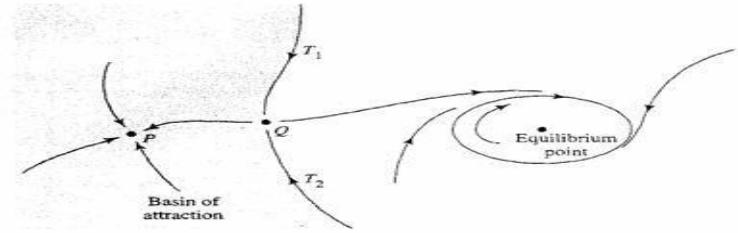


FIGURE 14.6 Illustration of the notion of a basin of attraction, and the idea of a separatrix.

Hyperbolic Attractors:

- A "hyperbolic attractor" is an attractor that possesses a "hyperbolic structure." This implies that the dynamics within the attractor exhibit certain properties related to expansion and contraction in different directions.
- In simpler terms, it refers to attractors that exist within, or whose properties can be better understood by, the use of hyperbolic geometry.
- **Key properties:**
  - They often display complex, sometimes chaotic, behavior.
  - They can be useful for modeling systems with hierarchical relationships.
  - They have properties like stable and unstable manifolds.

Manipulation of Attractors as a Recurrent Network Paradigm:

The manipulation of attractors as a recurrent network paradigm is a powerful concept in neural networks, particularly for tasks involving memory, pattern recognition, and decision-making. Here's a detailed breakdown:

Core Idea:

- The fundamental idea is to use attractors as stable states that represent specific patterns, memories, or decisions.
- By manipulating these attractors, the network can perform various cognitive functions.
- Recurrent neural networks (RNNs) are particularly well-suited for this paradigm because their recurrent connections allow them to maintain and manipulate internal states over time.

- **Attractor States as Representations:**

  - Each attractor state corresponds to a specific pattern or memory.

- The network's activity converges to an attractor when a related input is presented.

- **Attractor Basins:**
  - Each attractor has a basin of attraction, which is the region in state space from which the network will converge to that attractor.
  - The size and shape of the basins determine the network's robustness to noise and its ability to generalize.

- **Attractor Manipulation:**
  - This involves dynamically changing the network's attractors or its basins of attraction.
  - This can be achieved by:
    - Modifying the connection weights between neurons.
    - Applying external inputs that push the network towards a desired attractor.
    - Using control mechanisms that switch between different sets of attractors.

- **Recurrent Connections:**
  - Recurrent connections allow the network to maintain its internal state over time, which is essential for attractor dynamics.

Implementation Considerations:

- **Hopfield Networks:**
  - A classic example of attractor networks, but they have limitations in terms of capacity and stability.
- **Continuous-Time RNNs:**
  - These networks are well-suited for modeling continuous-time dynamics and can exhibit a wide range of attractor behaviors.
- **Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) Networks:**
  - These architectures can learn to manipulate attractors over long time scales, making them suitable for complex tasks.

Hopfield Models

Hopfield networks are a type of recurrent neural network that serve as a foundational model for understanding attractor network

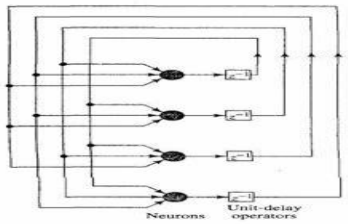


FIGURE 14.9 Architectural graph of a Hopfield network consisting of $N = 4$ neurons.

To study the dynamics of the Hopfield network, we use the neurodynamical model described in Eq. (14.16), which is based on the additive model of a neuron. Recognizing that $x_i(t) = \varphi_i(v_i(t))$, we may rewrite Eq. (14.16) in the form

$$C_j \frac{d}{dt} v_j(t) = -\frac{v_j(t)}{R_j} + \sum_{i=1}^{N} w_{ji} \varphi_i(v_i(t)) + I_j, \qquad j = 1, \ldots, N \tag{14.20}$$

To proceed with the discussion, we make the following assumptions:

1. The matrix of synaptic weights is *symmetric*, as shown by
$$w_{ji} = w_{ij} \quad \text{for all } i \text{ and } j \tag{14.21}$$
2. Each neuron has a *nonlinear* activation of its own—hence the use of $\varphi_i(\cdot)$ in Eq. (14.20).
3. The *inverse* of the nonlinear activation function exists, so we may write
$$v = \varphi_i^{-1}(x) \tag{14.22}$$

Let the sigmoid function $\varphi_i(v)$ be defined by the hyperbolic tangent function

$$x = \varphi_i(v) = \tanh\left(\frac{a_i v}{2}\right) = \frac{1 - \exp(-a_i v)}{1 + \exp(-a_i v)} \tag{14.23}$$

which has a slope of $a_i/2$ at the origin as shown by

$$\frac{a_i}{2} = \left.\frac{d\varphi_j}{dv}\right|_{v=0} \tag{14.24}$$

Henceforth we refer to a_i as the *gain* of neuron i.

The inverse output–input relation of Eq. (14.22) may thus be rewritten in the form

$$v = \varphi_i^{-1}(x) = -\frac{1}{a_i} \log\left(\frac{1 - x}{1 + x}\right) \tag{14.25}$$

The *standard* form of the inverse output–input relation for a neuron of unity gain is defined by

$$\varphi^{-1}(x) = -\log\left(\frac{1 - x}{1 + x}\right) \tag{14.26}$$

We may rewrite Eq. (14.25) in terms of this standard relation as

$$\varphi_i^{-1}(x) = \frac{1}{a_i} \varphi^{-1}(x) \tag{14.27}$$

Figure 14.10a shows a plot of the standard sigmoidal nonlinearity $\varphi(v)$, and Fig. 14.10b shows the corresponding plot of the inverse nonlinearity $\varphi^{-1}(x)$.

The energy (Lyapunov) function of the Hopfield network in Fig. 14.9 is defined by (Hopfield, 1984)

$$E = -\frac{1}{2} \sum_{j=1}^{N} \sum_{i=1}^{N} w_{ji} x_i x_j + \sum_{j=1}^{N} \frac{1}{R_j} \int_0^{x_j} \varphi_j^{-1}(x)\, dx - \sum_{j=1}^{N} I_j x_j \tag{14.28}$$

The energy function E defined by Eq. (14.28) may have a complicated *landscape* with many minima. The dynamics of the network are described by a mechanism that seeks out those minima.

Hence, differentiating E with respect to time, we get

$$\frac{dE}{dt} = -\sum_{j=1}^{N} \left(\sum_{i=1}^{N} w_{ji} x_i - \frac{v_j}{R_j} + I_j\right) \frac{dx_j}{dt} \tag{14.29}$$

The quantity inside the parentheses on the right-hand side of Eq. (14.29) is recognized as $C_j\, dv_j/dt$ by virtue of the neurodynamical equation (14.20). We may thus simplify Eq. (14.29) to

$$\frac{dE}{dt} = -\sum_{j=1}^{N} C_j \left(\frac{dv_j}{dt}\right) \frac{dx_j}{dt} \tag{14.30}$$

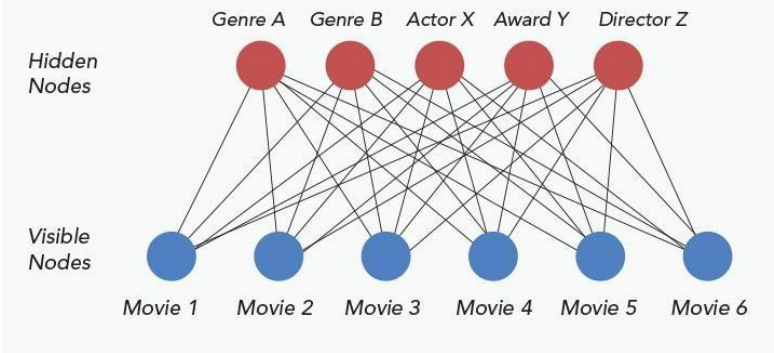
We now recognize the inverse relation that defines v_j in terms of x_j. The use of Eq. (14.22) in (14.30) yields

$$\begin{aligned} \frac{dE}{dt} &= -\sum_{j=1}^{N} C_j \left[\frac{d}{dt} \varphi_j^{-1}(x_j)\right] \frac{dx_j}{dt} \\ &= -\sum_{j=1}^{N} C_j \left(\frac{dx_j}{dt}\right)^2 \left[\frac{d}{dx_j} \varphi_j^{-1}(x_j)\right] \end{aligned} \tag{14.31}$$

From Fig. 14.10b we see that the inverse output–input relation $\varphi_j^{-1}(x_j)$ is a monotonically increasing function of the output x_j. It follows therefore that

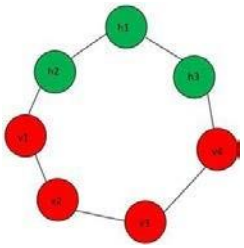
$$\frac{d}{dx_j} \varphi_j^{-1}(x_j) \geq 0 \quad \text{for all } x_j \tag{14.32}$$

Restricted Boltzmann Machine.



layer is only connected to neurons in the hidden layer, and vice versa. This allows the RBM to learn a compressed representation of the input data by reducing the dimensionality of the input.

The Boltzmann distribution (also known as Gibbs Distribution) which is an integral part of Statistical Mechanics and also explain the impact of parameters like Entropy and Temperature on the Quantum States in Thermodynamics. Due to this, it is also known as Energy-Based Models (EBM).



A restricted term refers to that we are not allowed to connect the same type layer to each other. In other words, the two neurons of the input layer or hidden layer can't connect to each other. Although the hidden layer and visible layer can be connected to each other.

As in this machine, there is no output layer so the question arises how we are going to identify, adjust the weights and how to measure that our prediction is accurate or not.

The energy function is given by

How do Restricted Boltzmann Machines work?

Let us consider an example in which we have some assumption that V1 visible unit activates the h1 and h2 hidden unit and V2 visible unit activates the h2 and h3 hidden. Now when any new visible unit let V5 has come into the machine and it also activates the h1 and h2 unit. So, we can back trace the hidden units easily and also identify that the characteristics of the new V5 neuron is matching with that of V1. This is because V1 also activated the same hidden unit earlier.

Applications of Restricted Boltzmann Machine

Restricted Boltzmann Machines (RBMs) have found numerous applications in various fields, some of which are:

- **Collaborative filtering:** RBMs are widely used in collaborative filtering

REAL LIFE EXAMPLES

UNIT – I : Introduction & Learning Process

1. Neural Network – Real-Time Example

Example: Google Photos face recognition
A neural network learns facial features (eyes, nose, mouth) from millions of images and automatically groups photos of the same person.

2. Human Brain vs Artificial Neural Network

Case Study: Stroke rehabilitation
Brain neurons rewire themselves after injury (plasticity). Similarly, neural networks adjust weights during learning to improve performance.

3. Models of a Neuron

Example: Email spam filter
Each word in an email is an input, weight represents importance, summation + activation decides spam or not.

4. Neural Networks as Directed Graphs

Example: Recommendation systems (Netflix)
Nodes = users, movies; edges = influence/weights. Direction shows data flow from input to output.

5. Network Architectures

Case Study: Autonomous vehicles

- CNN → image processing
- RNN → route prediction
- MLP → decision making

6. Knowledge Representation

Example: Chatbots
Weights store learned language patterns instead of explicit rules.

7. AI and Neural Networks

Case Study: Virtual assistants (Alexa, Siri)
Neural networks power speech recognition, NLP, and intent classification.

Learning Processes – Real-Time Examples

1. Error Correction Learning

Example: Handwritten digit recognition
Model compares predicted digit with actual digit and adjusts weights to reduce error.

2. Memory Based Learning

Example: Face unlock on smartphones
Stores previous face samples and compares with new input.

3. Hebbian Learning

Example: Brain learning habits
"Practice makes perfect" – neurons firing together strengthen connections.

4. Competitive Learning

Example: Customer segmentation
Neurons compete to represent different customer groups.

5. Boltzmann Learning

Case Study: Recommendation engines
Used to find optimal configurations in uncertain environments.

6. Credit Assignment Problem

Example: Chess A
Hard to decide which move caused win/loss → solved by backpropagation.

7. Adaptation & Statistical Nature

Example: Stock market prediction
Learning from noisy, uncertain data using probability distributions.

UNIT – II : Perceptrons & Multilayer Perceptrons

Single Layer Perceptron – Real-Time Examples

1. Adaptive Filtering Problem

Example: Noise cancellation in headphones
Perceptron adapts weights to cancel background noise.

2. Linear Least Square Filters

Example: Image smoothing
Minimizes difference between original and filtered image.

3. LMS Algorithm

Case Study: Echo cancellation in telephony
Continuously updates weights for changing environments.

4. Learning Curves

Example: Student learning graph
Error decreases as training increases.

5. Learning Rate Annealing

Example: Online shopping recommendations
Fast learning initially, slower fine-tuning later.

6. Perceptron Convergence Theorem

Case Study: Binary classification (pass/fail)
Guaranteed convergence if data is linearly separable.

7. Perceptron vs Bayes Classifier

Example: Medical diagnosis
Bayes handles uncertainty better when data is Gaussian.

Multilayer Perceptron (MLP)

1. XOR Problem

Case Study: Logic gate design
Single perceptron fails; MLP solves XOR using hidden layers.

2. Backpropagation Algorithm

Example: Speech recognition
Error propagated backward to adjust weights.

3. Heuristics

Example: Choosing number of hidden neurons
Based on trial-and-error to improve accuracy.

4. Output Representation & Decision Rule

Example: Digit recognition
Softmax output decides digit class.

5. Feature Detection

Case Study: Face detection
Hidden layers detect edges, shapes, faces.

UNIT – III : Back Propagation

1. Backpropagation & Differentiation

Example: Gradient descent in ML models
Calculates weight updates efficiently.

2. Hessian Matrix

Example: Faster training in deep learning
Used in second-order optimization.

3. Generalization

Case Study: Fraud detection
Model works well on unseen transactions.

4. Cross Validation

Example: ML competitions
Dataset split to avoid overfitting.

5. Network Pruning

Case Study: Mobile AI apps
Removes unnecessary neurons to save memory.

6. Virtues & Limitations

Virtue: High accuracy
Limitation: Slow convergence, local minima

7. Accelerated Convergence

Example: Momentum & Adam optimizer
Used in deep neural networks.

8. Supervised Learning

Example: Email classification
Uses labeled spam/non-spam data.

UNIT – IV : Self-Organizing Maps (SOM)

1. Feature Mapping Models

Example: Brain sensory maps
Visual cortex mapping similar patterns together.

2. SOM

Case Study: Market basket analysis
Groups similar customer behavior.

3. SOM Algorithm

Example: Color clustering in images
Similar colors mapped closer.

4. Properties of Feature Map

Example: Data visualization
High-dimensional data shown in 2D.

5. Computer Simulations

Example: Traffic pattern analysis
Simulated to detect congestion zones.

6. Learning Vector Quantization

Case Study: Speech recognition
Improves classification accuracy after SOM.

7. Adaptive Pattern Classification

Example: Handwriting recognition
Adapts to different writing styles.

UNIT – V : Neuro Dynamics & Hopfield Models

1. Dynamical Systems

Example: Weather prediction models
State evolves over time.

2. Stability of Equilibrium States

Example: Memory recall
Stable states represent stored memories.

3. Attractors

Example: Pattern completion
Partial image → complete image retrieval.

4. Neuro Dynamical Models

Case Study: Brain signal modeling (EEG)

5. Hopfield Networks

Example: Image restoration
Removes noise and restores patterns.

6. Restricted Boltzmann Machine (RBM)

Case Study: Deep belief networks
Used in feature learning and recommendation systems.

SUSTAINABLE DEVELOPMENT GOALS (SDGs)

Unit	Unit Title / Major Topics	Mapped SDGs	Justification (Course–SDG Alignment)
UNIT – I	ntroduction to Neural Networks, Human Brain analogy, Neuron models, Directed Graphs, Architectures, Knowledge Representation, AI & Neural Networks. Learning Processes: Error Correction, Memory-Based, Hebbian, Competitive, Boltzmann Learning, Credit Assignment, Adaptation, Statistical Learning	**SDG 4 – Quality Education** **SDG 9 – Industry, Innovation & Infrastructure**	Builds foundational AI knowledge, promotes technical education, and supports innovation in intelligent systems.
UNIT – II	single Layer Perceptrons: Adaptive Filtering, Linear Least Squares, LMS Algorithm, Learning Curves, Annealing Techniques, Perceptron Convergence Theorem, Bayes Classifier relation. Multilayer Perceptrons: Backpropagation, XOR Problem, Heuristics, Output Representation, Feature Detection	**SDG 9 – Industry, Innovation & Infrastructure** **SDG 8 – Decent Work & Economic Growth**	Enhances industrial automation and intelligent systems, improving productivity and economic growth.
UNIT – III	Backpropagation: Differentiation, Hessian Matrix, Generalization, Cross Validation, Network Pruning, Virtues & Limitations, Accelerated Convergence, Supervised Learnin	**Deepens supervised learning and optimization, fostering innovation and advanced AI education.**	Deepens supervised learning and optimization, fostering innovation and advanced AI education.
UNIT – IV	Self-Organizing Maps (SOM): Feature Mapping Models, SOM Algorithm, Properties, Simulations, Learning Vector Quantization (LVQ), Adaptive Pattern Classification	**SDG 11 – Sustainable Cities & Communities** **SDG 9 – Industry, Innovation & Infrastructure**	Enables unsupervised learning for clustering and pattern recognition, applicable in smart city solutions.

UNIT – V	Neurodynamics: Dynamical Systems, Stability of Equilibrium States, Attractors, Neurodynamical Models, Recurrent Network Paradigms, Hopfield Models, Restricted Boltzmann Machine (RBM)	**SDG 3 – Good Health & Well-Being** **SDG 9 – Industry, Innovation & Infrastructure**	Supports healthcare AI and optimization models, contributing to well-being and innovation.

THE END