Slide10

Haykin Chapter 14: Neurodynamics

CPSC 636-600
Instructor: Yoonsuck Choe
Spring 2008

Neural Networks with Temporal Behavior

- Inclusion of feedback gives temporal characteristics to neural networks: **recurrent networks**.
- Two ways to add feedback:
  - Local feedback
  - Global feedback
- Recurrent networks can become unstable or stable.
- Main interest is in recurrent network's **stability**: **neurodynamics**.
- Stability is a property of the *whole system*: coordination between parts is necessary.

Stability in Nonlinear Dynamical System

- **Lyapunov stablity**: more on this later.
- Study of neurodynamics:
  - Deterministic neurodynamics: expressed as nonlinear differential equations.
  - Stochastic neurodynamics: expressed in terms of stochastic nonlinear differential equations. Recurrent networks perturbed by noise.

Preliminaries: Dynamical Systems

- A **dynamical system** is a system whose state varies with time.
- **State-space model**: values of state variables change over time.
- Example: $x_1(t), x_2(t), ..., x_N(t)$ are state variables that hold different values under *independent variable* t. This describes a system of *order* N, and $\mathbf{x}(t)$ is called the *state vector*. The dynamics of the system is expressed using ordinary differential equations:

$$\frac{d}{dt}x_j(t) = F_j(x_j(t)), j = 1, 2, ..., N.$$

or, more conveniently

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{F}(\mathbf{x}(t)).$$

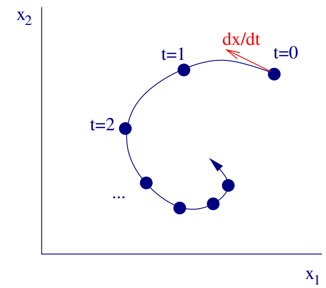
Autonomous vs. Non-autonomous Dynamical Systems

- Autonomous: $\mathbf{F}(\cdot)$ does not explicitly depend on time.
- Non-autonomous: $\mathbf{F}(\cdot)$ explicitly depends on time.

$\mathbf{F}$ as a Vector Field

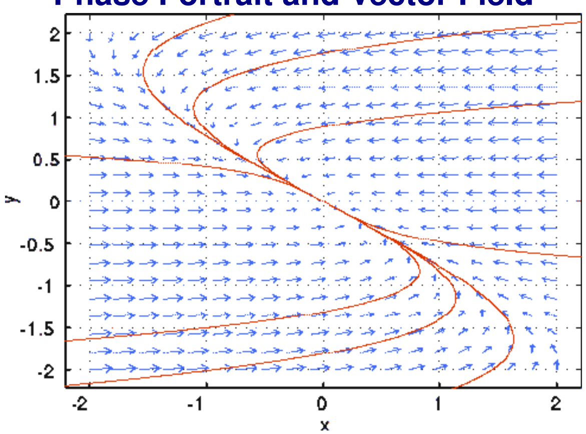
- Since $\frac{d\mathbf{x}}{dt}$ can be seen as velocity, $\mathbf{F}(\mathbf{x})$ can be seen as a velocity vector field, or a vector field.
- In a vector field, each point in space ($\mathbf{x}$) is associated with one unique vector (direction and magnitude). In a scalar field, one point has one scalar value.

State Space



- It is convenient to view the state-space equation $\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x})$ as describing the **motion** of a point in N-dimensional space (Euclidean or non-Euclidean). Note: t is continuous!
- The points traversed over time is called the **trajectory** or the **orbit**.
- The **tangent vector** shows the instantaneous velocity at the initial condition.

Phase Portrait and Vector Field



- Red curves show the state (phase) portrait represented by trajectories from different initial points.
- The blue arrows in the background shows the vector field.

Source: `http://www.math.ku.edu/~byers/ode/b_cp_lab/pict.html`

Conditions for the Solution of the State Space Equation

- A unique solution to the state space equation exists only under certain conditions, which restricts the form of $\mathbf{F}(\mathbf{x})$.
- For a solution to exist, it is sufficient for $\mathbf{F}(\mathbf{x})$ to be continuous in all of its arguments.
- For uniqueness, it must meet the **Lipschitz condition**.
- Lipschitz condition:
  - Let $\mathbf{x}$ and $\mathbf{u}$ be a pair of vectors in an open set $\mathcal{M}$ in a normal vector space. A vector function $\mathbf{F}(\mathbf{x})$ that satisfies:

    $$\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{u})\| \leq K\|\mathbf{x} - \mathbf{u}\|$$

    for some constant K, the above is said to be **Lipschitz**, and K is called the Lipschitz constant for $\mathbf{F}(\mathbf{x})$.
  - If $\partial F_i / \partial x_j$ are finite everywhere, $\mathbf{F}(\mathbf{x})$ meet the Lipschitz condition.

Stability of Equilibrium States

- $\bar{\mathbf{x}} \in \mathcal{M}$ is said to be an *equilibrium state* (or singular point) of the system if

$$\frac{d\bar{\mathbf{x}}}{dt} = \mathbf{F}(\bar{\mathbf{x}}) = \mathbf{0}.$$

- How the system behaves near these equilibrium states is of great interest.
- Near these points, we can approximate the dynamics by **linearizing** $\mathbf{F}(\mathbf{x})$ (using Taylor expansion) around $\bar{\mathbf{x}}$, i.e., $\mathbf{x}(t) = \bar{\mathbf{x}} + \Delta\mathbf{x}(t)$:

$$\mathbf{F}(\mathbf{x}) \approx \bar{\mathbf{x}} + \mathbf{A}\Delta\mathbf{x}(t)$$

where $\mathbf{A}$ is the Jacobian:

$$\mathbf{A} = \frac{\partial}{\partial \mathbf{x}}\mathbf{F}(\mathbf{x})\bigg|_{\mathbf{x}=\bar{\mathbf{x}}}$$

Stability of in Linearized System

- In the linearized system, the property of the Jacobian matrix $\mathbf{A}$ determine the behavior near equilibrium points.
- This is because

$$\frac{d}{dt}\Delta\mathbf{x}(t) \approx \mathbf{A}\Delta\mathbf{x}(t).$$

- If $\mathbf{A}$ is nonsingular, $\mathbf{A}^{-1}$ exists and this can be used to describe the **local** behavior near the equilibrium $\bar{\mathbf{x}}$.
- The eigenvalues of the matrix $\mathbf{A}$ characterize different classes of behaviors.

Eigenvalues/Eigenvectors

- For a square matrix $\mathbf{A}$, if a vector $\mathbf{x}$ and a scalar value λ exists so that

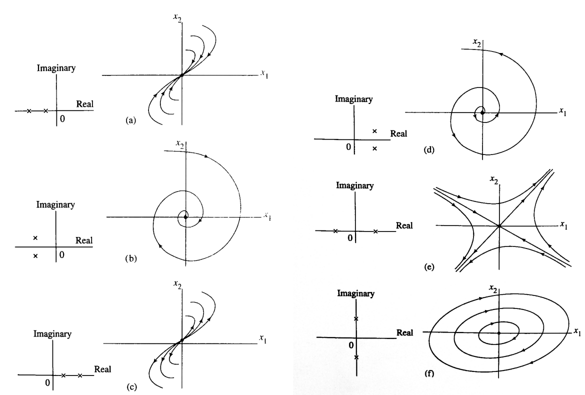
$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$$

then $\mathbf{x}$ is called an **eigenvector** of $\mathbf{A}$ and λ an **eigenvalue**.
- Note, the above is simply

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$$

- An intuitive meaning is: $\mathbf{x}$ is the direction in which applying the linear transformation $\mathbf{A}$ only changes the magnitude of $\mathbf{x}$ (by λ) but not the angle.
- There can be as many as n eigenvector/eigenvalue for an $n \times n$ matrix.

Example: 2nd-Order System



Positive/negative, real/imaginary character of **eigenvalues** of Jacobian determine behavior.

- Stable node (real -), unstable focus (Complex, + real)
- Stable focus (Complex, - real), Saddle point (real + -)
- Unstable node(real +), Center (Complex, 0 real)

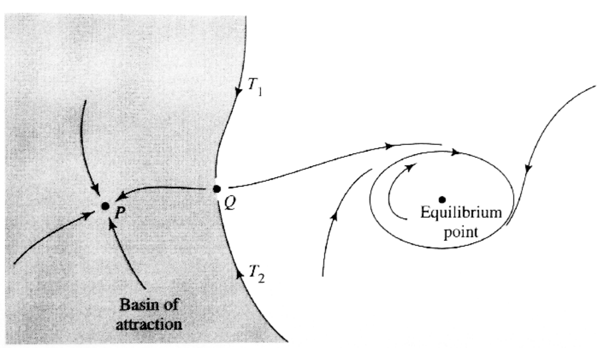
Definitions of Stability

- **Uniformly stable** for an arbitrary $\epsilon > 0$, if there exists a positive δ such that $\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$ implies $\|\mathbf{x}(t) - \bar{\mathbf{x}}\| < \epsilon$ for all $t > 0$.
- **Convergent** if there exists a positive δ such that $\|\mathbf{x}(0) - \bar{\mathbf{x}}\| < \delta$ implies $\mathbf{x}(t) \rightarrow \bar{\mathbf{x}}$ as $t \rightarrow \infty$
- **Asymptotically stable** if both stable and convergent.
- **Globally asymptotically stable** if stable and all trajectoreis of the system converge to $\bar{\mathbf{x}}$ as time t approaches infinity.

Lyapunov's Theorem

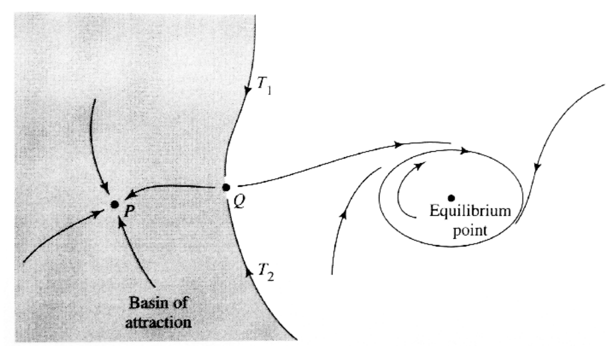
- **Theorem 1**: The equilibrium state $\bar{\mathbf{x}}$ is stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function $V(\mathbf{x})$ such that its derivative with respect to time is negative semidefinite in that region.
- **Theorem 2**: The equilibrium state $\bar{\mathbf{x}}$ is asymptotically stable if in a small neighborhood of $\bar{\mathbf{x}}$ there exists a positive definite function $V(\mathbf{x})$ such that its derivative with respect to time is negative definite in that region.
- A scalar function $V(\mathbf{x})$ that satisfies these conditions is called a **Lyapunov function** for the equilibrium state $\bar{\mathbf{x}}$.

Attractors



- Dissipative systems are characterized by attracting sets or manifolds of dimensionality lower than that of the embedding space. These are called **attractors**.
- *Regions* of initial conditions of nonzero state space volume *converge* to these attractors as time t increases.

Types of Attractors



- Point attractors (left)
- Limit cycle attractors (right)
- Strange (chaotic) attractors (not shown)

Neurodynamical Models

We will focus on state variables are continuous-valued, and those with dynamics expressed in differential equations or difference equations.

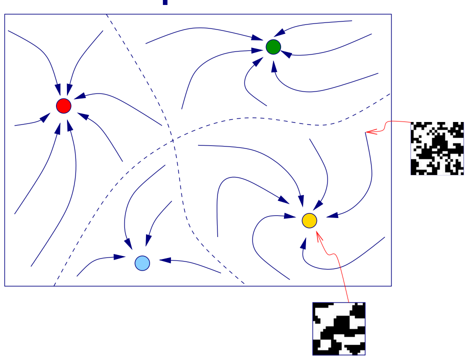
Properties:

- Large number of degree of freedom.
- Nonlinearity
- Dissipative (as opposed to conservative), i.e., open system.
- Noise

Manipulation of Attractors as a Recurrent Nnet Paradigm

- We can identify attractors with computational objects (associative memories, input-output mappers, etc.).
- In order to do so, we must exercise *control* over the location of the attractors in the state space of the system.
- A learning algorithm will manipulate the equations governing the dynamical behavior so that a desired location of attractors are set.
- One good way to do this is to use the **energy minimization** paradigm (e.g., by Hopfield).

Hopfield Model



- N units with full connection among every node (no self-feedback).
- Given M input patterns, each having the same dimensionality as the network, can be memorized in attractors of the network.
- Starting with an initial pattern, the dynamic will converge toward the attractor of the basin of attraction where the inital pattern was placed.

Discrete Hopfield Model

- Based on McCulloch-Pitts model (neurons with +1 or -1 output).
- Energy function is defined as

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ji}x_i x_j (i \neq j).$$

- Network dynamics will evolve in the direction that minimizes E.
- Implements a **content-addressable memory**.

Content-Addressable Memory

- Map a set of patterns to be memorized ξ_μ onto fixed points $\mathbf{x}_\mu$ in the dynamical system realized by the recurrent network.
- **Encoding**: Mapping from ξ_μ to $\mathbf{x}_\mu$
- **Decoding**: Reverse mapping from state space $\mathbf{x}_\mu$ to ξ_μ.

Hopfield Model: Storage

- The learning is similar to Hebbian learning:

$$w_{ji} = \frac{1}{N}\sum_{\mu=1}^{M} \xi_{\mu,j}\xi_{\mu,i}$$

with $w_{ji} = 0$ if $i = j$. (Learning is **one-shot**.)

- In matrix form the above becomes:

$$\mathbf{W} = \frac{1}{N}\sum_{\mu=1}^{M} \boldsymbol{\xi}_\mu \boldsymbol{\xi}_\mu^T - M\mathbf{I}$$

- The resulting weight matrix $\mathbf{W}$ is symmetric: $\mathbf{W} = \mathbf{W}^T$.

Hopfield Model: Activation (Retrieval)

- Initialize the network with a **probe pattern** $\boldsymbol{\xi}_{\text{probe}}$.

$$x_j(0) = \xi_{\text{probe},j}.$$

- Update output of each neuron (picking them by random) as

$$x_j(n+1) = \text{sgn}\left(\sum_{i=1}^{N} w_{ji}x_i(n)\right).$$

until $\mathbf{x}$ reaches a fixed point.

Spurious States

- The weight matrix $\mathbf{W}$ is symmetric, thus the eigenvalues of $\mathbf{W}$ are all real.
- For large number of patters M, the matrix is *degenerate*, i.e., several eigenvectors can have the same eigenvalue.
- These eigenvectors form a subspace, and when the associated eigenvalue is 0, it is called a *null space*.
- This is due to M being smaller than the number of neurons N.
- Hopfield network as content addressable memory:
  - Discrete Hopfield network acts as a vector projector (project probe vector onto subspace spanned by training patterns).
  - Underlying dynamics drive the network to converge to one of the corners of the unit hypercube.
- **Spurious states** are those corners of the hypercube that do not belong to the training pattern set.

Storage Capacity of Hopfield Network

- Given a probe equal to the stored pattern $\boldsymbol{\xi}_\nu$, the activation of the jth neuron can be decomposed into the signal term and the noise term:

$$\begin{aligned} v_j &= \sum_{i=1}^{N} w_{ji}\xi_{\nu,i} \\ &= \frac{1}{N}\sum_{\mu=1}^{M}\xi_{\mu,j}\sum_{i=1}^{N}\xi_{\mu,i}\xi_{\nu,i} \\ &= \underbrace{\xi_{\nu,j}}_{signal} + \underbrace{\frac{1}{N}\sum_{\mu=1,\mu\neq\nu}^{M}\xi_{\mu,j}\sum_{i=1}^{N}\xi_{\mu,i}\xi_{\nu,i}}_{noise} \end{aligned}$$

- The *signal-to-noise ratio* is defined as

$$\rho = \frac{\text{variance of signal}}{\text{variance of noise}} = \frac{1}{(M-1)/N} \approx \frac{N}{M}$$

- The reciprocal of ρ, called the *load parameter* is designated as α. According to Amit and others, this value needs to be less than 0.14 (critical value α_c).

Storage Capacity of Hopfield Network (cont'd)

- Given $\alpha = 0.14$, the storage capacity becomes

$$M_c = \alpha_c N = 0.14N$$

  when some error is allowed in the final patterns.

- For almost error-free performance, the storage capacity become

$$M_c = \frac{N}{2\log_e N}$$

- Thus, storage capacity of Hopfield network scales less than linearly with the size N of the network.
- This is a major limitation of the Hopfield model.

Cohen-Grossberg Theorem

- Cohen and Grossberg (1983) showed how to assess the stability of a certain class of neural networks:

$$\frac{d}{dt}u_j = a_j(u_j)\left[b_j(u_j) - \sum_{i=1}^{N} c_{ji}\varphi_i(u_i)\right], j = 1, 2, ..., N$$

.

- Neural network with the above dynamics admits a Lyapunov function defined as:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} c_{ji}\varphi_i(u_i)\varphi_j(u_j) - \sum_{j=1}^{N}\int_0^{u_j} b-j(\lambda)\varphi'_j(\lambda)d\lambda,$$

where

$$\varphi'(\lambda) = \frac{d}{d\lambda}(\varphi_j(\lambda)).$$

Cohen-Grossberg Theorem (cont'd)

- For the definition in the previous slide to be valid, the following conditions need to be met.
  - The synaptic weights are symmetric.
  - The function $a_j(u_j)$ satisfies the condition for *nonnegativity*.
  - The nonlinear activation function $\varphi_j(u_j)$ needs to follow the *monotonicity condition*:

$$\varphi'_j(u_j) = \frac{d}{du_j}\varphi_j(u_j) \geq 0.$$

- With the above

$$\frac{dE}{dt} \leq 0$$

  ensuring global stability of the system.

- Hopfield model can be seen as a special case of the Cohen-Grossberg theorem.

Demo

- Noisy input
- Partial input
- Capacity overload