Unit 5: Multilayer Perceptron

Nonlinear Filtering in Neural Networks: From Static to Dynamic Pattern Recognition

Kiran Bagale

St. Xavier's College

August 4, 2025

Pattern Recognition Types

Structural Pattern Recognition

- Static neural networks
- Multilayer perceptrons
- Time-independent processing
- Response depends only on current input

Temporal Pattern Recognition

- Dynamic neural networks
- **Nonlinear filtering**
- Time-dependent processing
- Response depends on current **and past** inputs

Key Insight

Time is an ordered quantity that constitutes an important ingredient of the learning process in temporal-pattern-recognition tasks.

Requirements for Dynamic Networks

Add Memory

Static Neural Network $\longrightarrow$ Dynamic Neural Network

Current Input Only

Current + Past Inputs

Essential Component: Short-term Memory

A neural network must be given **short-term memory** in one form or another to become dynamic.

Implementation Method: Time Delays

- At the synaptic level (inside the network)
- At the input layer (external to the network)
- **Neurobiologically motivated** - signal delays are omnipresent in the brain

Two Basic Approaches

Implicit Representation

- Time represented by its **effect** on signal processing
- Digital implementation approach:
  1. Uniform sampling of input signal
  2. Convolution of synaptic weights with input samples
  3. Temporal structure → Spatial structure

Explicit Representation

- Time has its own **particular representation**
- Example: Bat echolocation system
  - FM signal emission
  - Multiple frequency comparisons
  - Delay line matching for range estimation

Implicit vs Explicit: Visual Comparison

Implicit Representation

Input Signal (time series)

↓

Spatial Network Structure

Temporal → Spatial

Explicit Representation

Input + Time Info

↓

Network with Time Delays

Direct Time Encoding

Biological Motivation

Neurobiological Foundation

Signal Delays in the Brain

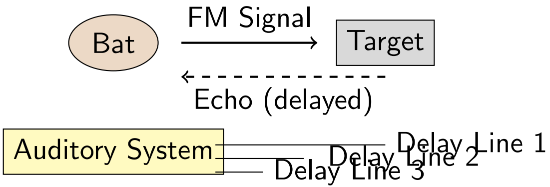
- Signal delays are **omnipresent** in biological neural systems
- Play an **important role** in neurobiological information processing
- Well-established research foundation (Braitenberg, 1967, 1977, 1986; Miller, 1987)

Case Study:

Bat Echolocation

1. Bat emits short frequency-modulated (FM) signal
2. Same intensity maintained for each frequency channel
3. Multiple frequency comparisons via auditory receptor array
4. Delay line matching provides range estimation
5. Unknown echo delay → matching neuron responds

Bat Echolocation System



Range Estimation

Nonlinear Filter Structure

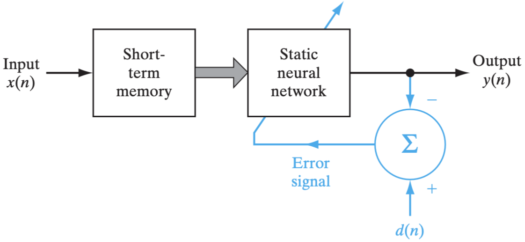


Figure: Nonlinear filter built on a static neural network.

Key Components

- **Cascade connection** of two subsystems
- **Clear separation** of processing roles:
  - a. Static network: accounts for nonlinearity b. Memory: accounts for time
- **SIMO structure**: Single-input, multiple-output memory

Discrete-Time Memory Structure

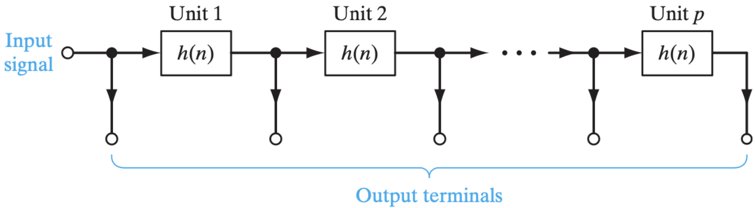


Figure: Caption

Tapped-Delay-Line Memory [redacted]

- p identical sections connected in cascade
- Each section characterized by impulse response $h(n)$
- Memory order: p (number of sections)
- Output terminals: $p + 1$ (including direct connection)

Theorem Properties and Applications

Key Properties

- **Shift-invariant**: Output shift corresponds to input shift
- **Myopic**: "Uniformly fading memory" - distant past has less influence
- **Causal**: Output at time $n \geq 0$ only depends on inputs at $n = 0$
- **Universal approximation**: Any such map can be approximated

Practical Applications

- **Time series prediction**: Build predictive models using $y(n) = f(x(n-1), x(n-2), \ldots)$
- **System identification**: Model underlying nonlinear physical laws
- **Stability guarantee**: Structure is inherently stable if linear filters are stable
- **Linear output neurons**: Recommended to avoid amplitude limitations

Tapped-Delay-Line Memory

Generating Kernel

$$h(n) = \delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

Overall Impulse Response

$$h_{\text{overall}}(n) = \delta(n - p) = \begin{cases} 1, & n = p \\ 0, & n \neq p \end{cases}$$

Characteristics

- Memory depth: $D = p$
- Memory resolution: $R = 1$
- Depth-resolution product: $D \times R = p$
- **High resolution, finite depth**

Gamma Memory

Generating Kernel

$h(n) = \mu(1-\mu)^{n-1}, \quad n \geq 1$ where $0 < \mu < 2$ for stability.

Overall Impulse Response

$h_{\text{overall}}(n) = \binom{n-1}{p-1}\mu^p(1-\mu)^{n-p}, \quad n \geq p$

Characteristics

- Memory depth: $D = p/\mu$
- Memory resolution: $R = \mu$
- Depth-resolution product: $D \times R = p$
- **Tunable depth-resolution tradeoff**

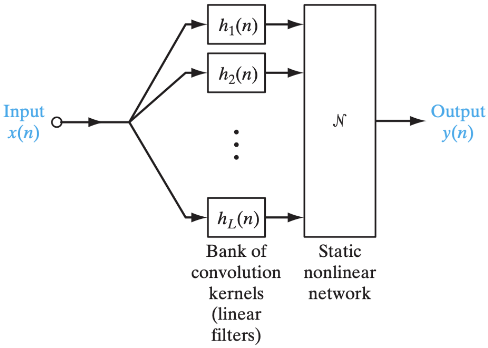


Figure: Structure of Myopic Mapping

Universal Myopic Mapping Theorem

Any shift-invariant myopic dynamic map can be uniformly approximated arbitrarily well by a structure consisting of two functional blocks: a bank of linear filters feeding a static neural network.

NETtalk System Architecture

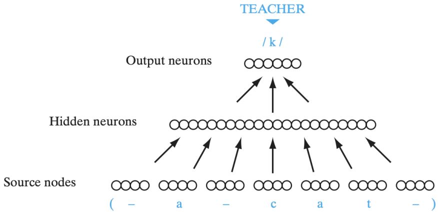


Figure: Schematic diagram of the NETtalk network architecture.

NETtalk Specifications

- **Architecture**: 203 input nodes, 80 hidden neurons, 26 output neurons
- **Task**: Convert English text to phonemes
- **Context window**: 7 letters (target letter $\pm$ 3 context)
- **Training**: Backpropagation with 18,629 weights

NETtalk Performance Characteristics

Human-Like Learning Patterns

- **Power law learning**: Training followed a power law progression
- **Generalization**: Better performance with more training words
- **Graceful degradation**: Slow performance decline with damaged connections
- **Fast relearning**: Recovery after damage was faster than original training

Significance

NETtalk was the **first demonstration** of a massively parallel distributed network that:

- Successfully converted English speech to phonemes
- Started with "innate" knowledge of input patterns
- Gradually acquired competence through practice
- Exhibited human-like learning characteristics

Key Takeaways

Nonlinear Filter Architecture

- **Cascade structure**: Short-term memory + static neural network
- **Clear role separation**: Memory handles time, network handles nonlinearity
- **SIMO configuration**: Multiple delayed versions of input signal

Universal Myopic Mapping Theorem

- **Theoretical foundation**: Any shift-invariant myopic map can be approximated
- **Practical structure**: Bank of linear filters + static nonlinear network
- **Stability guarantee**: Inherently stable if linear components are stable

Memory Structures

- **Tapped-delay-line**: High resolution, finite depth ($D \times R = p$)
- **Gamma memory**: Tunable depth-resolution tradeoff (parameter)
- **Design choice**: Depends on application requirements

Statistical and Computational Distinctions in Supervised Learning

- Statistical issues: Structural Risk Minimization (SRM)
- Computational issues: Time complexity constraints
- Trade-offs between approximation, estimation, and optimization errors

The Fundamental Question

Key Question in Supervised Learning

Does a training sample consisting of N independent and identically distributed examples

$$(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \ldots, (\mathbf{x}_N, d_N)$$

contain sufficient information to construct a learning machine capable of good generalization performance?

- Answer lies in **Structural Risk Minimization** (Vapnik, 1982, 1998)
- Distinguishes between different scales of learning problems
- Determines appropriate optimization strategies

Definitions: Problem Scale Classification

Definition I: Small-Scale Learning (Bottou, 2007)

A supervised-learning problem is said to be of a **small-scale** kind when the **size of the training sample** (i.e., the number of examples) is the active budget constraint imposed on the learning process.

Definition II: Large-Scale Learning (Bottou, 2007)

A supervised-learning problem is said to be of a **large-scale** kind when the **computing time** is the active budget constraint imposed on the learning process.

The **active budget constraint** *distinguishes one learning problem from the other.*

Illustrative Examples

Small-Scale Example

Adaptive Equalizer

- Compensates for channel distortion
- Uses LMS algorithm
- Based on stochastic gradient descent
- Limited by training data size

Large-Scale Example

Check Reader System

- Processes {image, amount} pairs
- Field & character segmentation
- Character recognition
- Syntactical interpretation
- Uses convolutional networks
- Deployed since 1996 (billions of checks)

Small-Scale Learning Problems

Available Design Variables

Three variables available to the designer:

1. Number of training examples, N
2. Permissible size K of the family of approximating network functions $\mathcal{F}$
3. Computational error ϵ introduced in optimization

Design Options (Active constraint: training sample size)

- **Reduce estimation error**: Make N as large as budget permits
- **Reduce optimization error**: Set $\epsilon \to 0$ (i.e., $\mathbf{w} \to \hat{\mathbf{w}}_N$)
- **Adjust family size**: Set $|\mathcal{F}|$ to reasonable extent

Solution: Structural Risk Minimization with approximation-estimation tradeoff

Large-Scale Learning Problems

Active Budget Constraint

Computing time T is the limiting factor, leading to **more complicated trade-offs**

Excess Error Decomposition

$$J(\mathbf{w}) - J(\hat{f}^*) = \underbrace{J(\mathbf{w}) - J(\hat{\mathbf{w}}_N)}_{\text{Optimization error}} + \underbrace{J(\hat{\mathbf{w}}_N) - J(\mathbf{w}_N^*)}_{\text{Estimation error}} + \underbrace{J(\mathbf{w}_N^*) - J(\hat{f}^*)}_{\text{Approximation error}}$$

- **New term:** Optimization error (related to computational error ϵ)
- Last two terms common to both problem scales
- Must account for computing time T in all decisions

Trade-offs in Large-Scale Problems

Optimization Challenge

Minimize the sum of three error terms by adjusting:

- Number of examples, N
- Permissible size K of approximating functions, $\mathcal{F}_K$
- Computational error ϵ (no longer zero)

Complexity

Computing time T depends on **all three variables** $(N, \mathcal{F}, \epsilon)$

Consequence

Reducing ϵ (to decrease optimization error) requires increasing N or $\mathcal{F}$ or both, which adversely affects approximation and estimation errors.

Optimization Algorithm Categories

Algorithm Classification

Performance characterized by $\log \epsilon$ versus $\log T$ plots:

Category	**Algorithm Type**	**Example**
Bad	Stochastic Gradient Descent	On-line learning
Mediocre	Gradient Descent	Batch learning
Good	Second-order methods	BFGS, Quasi-Newton

- Algorithm choice determines final trade-offs
- Second-order methods offer better computational efficiency
- Stochastic methods may be preferred for very large datasets

Analysis Approach: Bounds vs Convergence Rates

Small-Scale Problems

- VC theory provides bounds
- Approximation error bounds well understood
- Constants in formulas are reasonable
- Structural Risk Minimization adequate

Large-Scale Problems

- VC theory constants are poor
- Bounds less practical
- **Convergence rates** more productive
- Focus on exponents of error decrease

Key Insight

For large-scale problems, analyze how errors decrease as ϵ decreases and both $\mathcal{F}$ and N increase, considering computational time T growth.

Summary and Current State

Key Distinctions

- **Small-scale**: Limited by training data size
- **Large-scale**: Limited by computational resources
- Different optimization strategies required
- Trade-offs become more complex for large-scale problems

Current Research Status

"Whereas the study of small-scale learning problems is well-developed, the study of large-scale learning problems is in its early stages of development."

- Small-scale: Mature theory (SRM, VC theory)
- Large-scale: Emerging field with active research
- Algorithm choice crucial for large-scale success