Unit 5 Multilayer Perceptrons

Virtues and Limitations of Back-Propagation Learning

Kiran Bagale

August 2025

Outline

Back-Propagation

Definition

The back-propagation algorithm is a **computationally efficient technique** for computing the gradients (i.e., first-order derivatives) of the cost function $\mathcal{E}(w)$, expressed as a function of the adjustable parameters (synaptic weights and bias terms) that characterize the multilayer perceptron.

Important Note

Back-propagation is **NOT** an algorithm intended for the optimum design of a multilayer perceptron. Rather, it is a technique for gradient computation.

Computational Power Sources

The computational power of the back-propagation algorithm is derived from two distinct properties:

1. The back-propagation algorithm is **simple to compute locally**.
2. It performs **stochastic gradient descent** in weight space, when the algorithm is implemented in its on-line (sequential) mode of learning.

Key Insight

These properties make back-propagation both computationally feasible and theoretically sound for training multilayer networks.

The Parity Function Problem

Parity Function Definition

$$\psi_{PARITY}(X) = \begin{cases} 1 & \text{if } |X| \text{ is an odd number} \\ 0 & \text{otherwise} \end{cases}$$

- The order equals the number of inputs
- Time required scales **exponentially** with number of inputs
- Projections suggest back-propagation may be overly optimistic for complex functions

Scaling Challenge

For large-scale, real-world problems, careful architectural design is crucial to successful application of back-propagation learning.

Computational Complexity

Linear Complexity

The computational complexity of the back-propagation algorithm is **linear in** W; that is, it is $O(W)$.

All computations in both forward and backward passes are linear in the synaptic weights:

- **Forward pass**: Induced local fields computation
- **Backward pass**: Local gradients and weight updates

This linearity holds regardless of where the synaptic weight appears in the chain of computations.

Sensitivity Analysis

Sensitivity Definition

The sensitivity of an input-output mapping function F with respect to parameter ω is:

$$S_\omega^F = \frac{\partial F / F}{\partial \omega / \omega}$$

Key Benefits:

- Efficient computation of partial derivatives for all weights
- Complexity is linear in W (total number of weights)
- Works regardless of weight position in the network

Robustness Properties

H-Optimal Filtering

- **Linear case**: LMS algorithm is H-optimal (minimizes maximum energy gain)
- **Nonlinear case**: Back-propagation is **locally** H-optimal

Local Optimality

The term "local" means the initial weight vector must be sufficiently close to the optimum value to avoid getting trapped in poor local minima.

Both LMS and back-propagation belong to the same class of H-optimal filters.

Function Approximation

Nested Sigmoidal Structure

For a single output, the multilayer perceptron manifests as:

$$F(x, w) = \varphi\left(\sum_k w_{ok}\varphi\left(\sum_j w_{kj}\varphi\left(\cdots\varphi\left(\sum_i w_{ji}x_i\right)\right)\right)\right)$$

Key Properties:

- $\varphi(\cdot)$ is sigmoid activation function
- w denotes entire set of synaptic weights
- Forms a **universal approximator**
- Unusual structure in classical approximation theory

Local Minima Problem

The Challenge

The error surface contains **local minima** (isolated valleys) in addition to global minima. Back-propagation, being a hill-climbing technique, risks getting trapped.

Consequences:

- Network may get stuck in local minimum
- Local minimum may be far above global minimum
- Difficult to determine numbers of local vs. global minima
- Learning process may terminate prematurely

Solution

Consider optimally annealed on-line learning algorithm (Section 4.10).

Convergence Issues

Stochastic Nature

Back-propagation uses "instantaneous estimates" for gradients, making it **stochastic** in nature.

Two fundamental causes of slow convergence:

1. **Flat error surface**: Small derivative magnitude leads to small weight adjustments, requiring many iterations.
2. **Wrong direction**: The negative gradient vector may point away from the minimum, causing movement in wrong direction.

The algorithm tends to "zigzag" around the true direction to minimum.

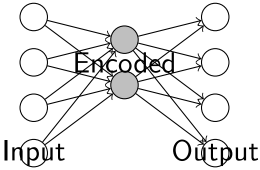
Data Compression System

Replicator Network Structure:

- Input and output layers: same size m
- Hidden layer size $M < m$
- Fully connected network
- Identity mapping objective

Training Process:

- Pattern x applied to input
- Desired output $=$ input pattern
- Network learns to encode/decode
- Compression ratio depends on M/m



Connectionist Paradigm

Locality Constraint

Back-propagation exemplifies the **connectionist paradigm** that relies on local computations to discover information-processing capabilities.

Three principal reasons for local computations:

1. Neural networks with local computations serve as **metaphors for biological neural networks**.
2. Local computations provide **graceful degradation** in performance from hardware errors, enabling **fault-tolerant network design**.
3. Local computations favor **parallel architectures** for efficient neural network implementation.

Replicator (Identity) Mapping

Feature Detection Role

Hidden neurons in back-propagation trained multilayer perceptrons play a critical role as **feature detectors**.

Applications:

- **Encoder**: Hidden layer produces compressed representation
- **Decoder**: Reconstructs original input from compressed form
- **Data compression**: Effective when $M \ll m$
- **Feature extraction**: Hidden units learn important patterns

Design Strategy

This approach is illustrated for optical character recognition problems, where architectural constraints incorporate prior knowledge about the task.

Summary: Virtues of Back-Propagation

Key Strengths

- **Computational efficiency**: $O(W)$ complexity
- **Universal approximation**: Can approximate any continuous function
- **Sensitivity analysis**: Efficient gradient computation
- **Robustness**: Locally H-optimal properties
- **Versatility**: Applicable to various architectures

Summary: Limitations of Back-Propagation

Key Challenges

- **Local minima**: Risk of suboptimal solutions
- **Slow convergence**: Stochastic nature causes zigzagging
- **Scaling issues**: Exponential complexity for some problems
- **Architecture dependence**: Requires careful design for large problems
- **No global optimality guarantee**: May not find best solution

Final Thoughts

Balance of Trade-offs

Back-propagation remains a fundamental algorithm in neural networks despite its limitations. Success depends on:

- Careful architectural design
- Appropriate problem selection
- Understanding of algorithm limitations
- Use of complementary techniques when needed

Thank you for your attention!