Unit 5 Multilayer Perceptron: Optimization Methods

Supervised Learning Viewed as an Optimization Problem

Kiran Bagale

July, 2025

Overview

What is Neural Network Training?

- **Goal**: Find the best set of weights that minimize prediction errors
- **Challenge**: The error landscape is complex and non-linear
- **Approach**: Use mathematical optimization techniques
- **Key Insight**: We're searching through a multi-dimensional space of possible solutions

The Optimization Problem

Training a neural network is essentially finding the optimal point on an error surface where our model performs best.

Understanding the Error Surface

Key Concepts:

- **Error Surface**: A landscape showing how prediction errors change with different weight values
- **Gradient**: The slope or direction of steepest increase
- **Local Minima**: Valleys that might trap our optimization
- **Global Minimum**: The deepest valley (best solution)

Imagine hiking down a mountainous terrain to find the lowest point - that's essentially what we're doing in neural network training!

Gradient-Based Learning: The Foundation

Basic Idea

Move in the direction opposite to the gradient (steepest descent) to minimize errors.

Advantages:

- Simple to understand and implement
- Computationally efficient for each step
- Works well for many practical problems

Limitations:

- Can be slow to converge
- May get stuck in poor local solutions
- Sensitive to the learning rate choice

Beyond Gradients: Using Curvature Information

The Next Level

Instead of just knowing which way is downhill, also consider how curved the landscape is.

Why Curvature Matters:

- Helps distinguish between steep cliffs and gentle slopes
- Enables more intelligent step size selection
- Can lead to faster convergence
- Provides better understanding of the optimization landscape

The Trade-off:

- More computation per step
- Higher memory requirements
- More complex implementation

Newton's Method: The Ideal (But Impractical)

Theoretical Advantages:

- Optimal convergence for well-behaved problems
- Can reach the solution in one step for simple cases
- Uses complete curvature information

Practical Problems:

- **Computational Cost**: Requires inverting large matrices
- **Numerical Issues**: Matrix inversions can be unstable
- **Limited Scope**: Only works well for specific problem types

Reality Check

Pure Newton's method is rarely used in practice due to these computational challenges.

Conjugate Gradient: Smart Compromise

The Middle Ground

Faster than simple gradient descent, more practical than Newton's method.

Key Innovation: Non-interfering Directions

- Each search direction doesn't undo progress from previous directions
- Like choosing orthogonal paths in a transformed space
- Systematically explores the solution space

Practical Benefits:

- Guaranteed convergence in finite steps (for quadratic problems)
- No matrix inversions required
- Moderate computational requirements

How Conjugate Gradient Works

1. **Start**: Choose initial weights and compute error gradient
2. **Search**: Move along a carefully chosen direction
3. **Optimize**: Find the best step size in that direction
4. **Update**: Choose next direction that won't interfere with previous progress
5. **Repeat**: Continue until convergence

Two Popular Variants

- **Fletcher-Reeves**: More conservative, stable
- **Polak-Ribière**: Often faster, preferred for neural networks

Line Search: Finding the Right Step Size

The Challenge: Once we know which direction to go, how far should we step?

The Solution: Line Search

- **Bracketing**: Find a range containing the optimal step size
- **Refinement**: Narrow down to the best step size
- **Interpolation**: Use curve fitting to estimate the minimum

Real-World Analogy

Like adjusting your stride when walking downhill - too small and you move slowly, too large and you might overshoot and go uphill again.

Quasi-Newton: Approximating the Ideal

Core Idea

Build an approximation to Newton's method that's computationally practical.

How It Works:

- Gradually build up curvature information from gradient observations
- Maintain a running approximation of the inverse curvature matrix
- Update this approximation efficiently at each step

Popular Algorithms:

- **BFGS**: Considered the gold standard
- **DFP**: Historical importance, less commonly used today

Quasi-Newton Trade-offs

When Quasi-Newton Excels:

- Small to medium-sized neural networks
- Problems where high accuracy is needed
- Situations with expensive function evaluations

When to Avoid:

- Very large networks (memory constraints)
- Online learning scenarios
- When gradient computation is cheap

Memory Consideration

Quasi-Newton methods store an approximation matrix, requiring significantly more memory than gradient-only methods.

Levenberg-Marquardt: Best of Both Worlds

The Hybrid Approach

Smoothly transitions between gradient descent and Newton's method based on local conditions.

Adaptive Behavior:

- **Far from optimum**: Acts like gradient descent (safe, stable)
- **Near optimum**: Acts like Newton's method (fast convergence)
- **Automatic switching**: Uses a damping parameter to control the transition

Particularly Effective For:

- Regression problems (least squares)
- Small to medium networks
- Problems where robustness is important

The Damping Strategy:

1. Try a step with current damping parameter
2. If error decreases: Accept step, reduce damping (move toward Newton)
3. If error increases: Reject step, increase damping (move toward gradient descent)
4. Repeat until convergence

Driving Analogy

Like a car's automatic transmission - shifts to the appropriate "gear" (method) based on current conditions without driver intervention.

Second-Order Stochastic Methods

The Challenge: Online learning with second-order efficiency

Approximation Strategies:

- **Diagonal Approximation**: Keep only the main diagonal of curvature information
- **Low-rank Approximation**: Use matrix decomposition techniques
- **Block-wise Updates**: Update curvature information in chunks

Benefits:

- Faster convergence than standard stochastic gradient descent
- More practical than full second-order methods
- Suitable for large-scale problems

Choosing the Right Method

Method	Network Size	Convergence	Memory
Gradient Descent	Any	Slow	Low
Conjugate Gradient	Medium-Large	Moderate	Low
Quasi-Newton (BFGS)	Small-Medium	Fast	High
Levenberg-Marquardt	Small-Medium	Fast	Medium

Selection Guidelines:

- **Small networks**: Quasi-Newton or Levenberg-Marquardt
- **Large networks**: Conjugate Gradient methods
- **Online learning**: Enhanced stochastic methods
- **Robustness critical**: Levenberg-Marquardt

Computational Complexity Considerations

Per-iteration Costs:

- **First-order methods**: Linear in number of weights
- **Second-order methods**: Quadratic in number of weights
- **Approximation methods**: Between linear and quadratic

The Fundamental Trade-off

Faster convergence per iteration vs. more computation per iteration

Total Training Time Depends On:

- Network architecture
- Problem complexity
- Required accuracy
- Available computational resources

Implementation Challenges

Common Issues:

- **Numerical Stability**: Matrix operations can be sensitive
- **Convergence Criteria**: When to stop the optimization
- **Initialization**: Starting points affect final solutions
- **Hyperparameter Tuning**: Learning rates, damping parameters, etc.

Practical Solutions:

- Use robust numerical libraries
- Implement multiple stopping criteria
- Try several initialization strategies
- Use adaptive parameter adjustment

Modern Perspective

Historical Importance:

- These methods laid the foundation for neural network optimization
- Many principles still apply to modern deep learning
- Understanding these helps with debugging and method selection

Current Relevance:

- Still used for smaller, specialized networks
- Principles influence modern optimizers (Adam, RMSprop, etc.)
- Important for understanding optimization landscapes

Key Insight

While modern deep learning often uses simpler methods, understanding these classical approaches provides crucial optimization intuition.

Summary

Key Takeaways:

1. Neural network training is fundamentally an optimization problem
2. Different methods make different trade-offs between speed, memory, and reliability
3. Second-order information can dramatically improve convergence
4. Method selection should consider network size and computational constraints
5. Understanding classical methods provides insight into modern optimizers

The art of neural network training lies in choosing the right optimization strategy for your specific problem!

Questions & Discussion

Thank you for your attention!

Questions and Discussion

Understanding optimization is the key to successful neural network training.