Day 30: From Second-Order Methods to Practical Tricks

Bridging Theory and Practice in Deep Learning Optimization

1. The Fundamental Question

Do we really need the Hessian in deep learning?

While second-order methods offer theoretical advantages with quadratic convergence rates, practical constraints demand innovative approximations that balance computational efficiency with curvature awareness.

2. The Computational Reality: Why Exact Hessian Fails

Prohibitive Costs for Modern Networks

- **Memory:** $O(n^2)$ storage $\rightarrow$ ResNet-50 (25M params) needs **500 TB**
- **Computation:** $O(n^3)$ inversion $\rightarrow$ GPT-3 (175B params) is **computationally impossible**
- **Numerical Issues:** Condition numbers often exceed $\kappa > 10^6$

3. Practical Alternatives: The Smart Compromises

3.1. Quasi-Newton Methods (L-BFGS)

$$H_{k+1}^{-1} = (I - \rho_k s_k y_k^T) H_k^{-1} (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

where $\rho_k = \frac{1}{y_k^T s_k}$, $s_k = \theta_{k+1} - \theta_k$, $y_k = \nabla L_{k+1} - \nabla L_k$

L-BFGS Performance

- **Memory:** $O(mn)$ vs $O(n^2)$ for exact Hessian ($m = 5 - 20$)
- **Speed:** Superlinear convergence in practice
- **Limitations:** Sensitive to batch size and hyperparameters

3.2. Adaptive Optimizers: Diagonal Preconditioning

Adam (Most Popular: Used in 70% of Papers):

$$m_t = 0.9m_{t-1} + 0.1g_t, \quad v_t = 0.999v_{t-1} + 0.001g_t^2$$

$$\theta_{t+1} = \theta_t - \frac{0.001}{\sqrt{v_t} + 10^{-8}} m_t$$

Optimizer Performance Comparison

Method	Memory	Convergence	Use Cases
Newton	$O(n^2)$	Quadratic	Theoretical
L-BFGS	$O(mn)$	Superlinear	Small batches
Adam	$O(n)$	Fast	Default choice
SGD+Momentum	$O(n)$	Linear	Computer Vision
SGD	$O(n)$	Slow	Simple models

4. Momentum: Physical Intuition for Curvature

4.1. Classical Momentum ($\gamma = 0.9$)

$$v_t = 0.9v_{t-1} + \eta g_t, \quad \theta_{t+1} = \theta_t - v_t$$

4.2. Nesterov Accelerated Gradient

$$v_t = 0.9v_{t-1} + \eta \nabla L(\theta_t - 0.9v_{t-1})$$

Momentum Benefits

- **Acceleration:** 10-30% faster convergence than vanilla SGD
- **Stability:** Reduces oscillation in ill-conditioned landscapes
- **Convergence:** $O(1/t^2)$ vs $O(1/t)$ for gradient descent

5. Regularization: Implicit Curvature Control

5.1. Weight Decay (L_2 Regularization)

$$L_{\text{reg}} = L(\theta) + 0.0005\|\theta\|_2^2$$

Adds $0.0005I$ to Hessian, improving conditioning: $\lambda_i^{\text{reg}} = \lambda_i + 0.0005$

5.2. Practical Learning Rate Strategies

- **Adam:** 10^{-3} to 10^{-5} (default: 0.001)
- **SGD:** 10^{-1} to 10^{-3} with momentum 0.9
- **Warmup:** Critical for batches > 1024 samples
- **Cosine Decay:** Often outperforms step decay by 1-2%

6. Modern Innovations and Practical Guidelines

Optimizer Selection Strategy

1. **Start with Adam** for most architectures (70% success rate)
2. **Try SGD+Momentum** if generalization is priority (especially CV)
3. **Consider L-BFGS** for small, deterministic problems
4. **Use advanced methods** (K-FAC, Shampoo) only with sufficient resources

7. Key Insight: The Spirit of Second-Order Methods Lives On

While deep learning abandoned exact Hessian methods due to computational constraints, modern optimizers successfully incorporate second-order principles through:

- **Adaptive learning rates** that approximate diagonal Hessian information
- **Momentum** that mimics curvature-aware navigation
- **Regularization** that implicitly controls Hessian conditioning
- **Architectural choices** (BatchNorm, residuals) that improve landscape geometry

8. Empirical Evidence: Why Approximations Work

Performance Statistics

- Adam typically achieves 2-5x faster convergence than SGD
- Well-tuned SGD+Momentum often provides best generalization
- L-BFGS can reach similar accuracy in 50-70% fewer iterations
- Adaptive methods reduce sensitivity to learning rate by 10-100x

9. The Bottom Line: Practical Wisdom

Actionable Recommendations

- Use Adam as your default optimizer (learning rate = 0.001)
- For vision tasks, try SGD with momentum (lr=0.1, momentum=0.9)
- Always use learning rate warmup for large batch training
- Monitor gradient norms to detect ill-conditioning
- Regularize with weight decay (typically 0.0001 to 0.001)

10. Conclusion: Theory Meets Practice

The evolution from Newton's method to modern adaptive optimizers represents a perfect case study of theoretical insight driving practical innovation. Today's methods successfully balance:

- **Efficiency:** $O(n)$ complexity instead of $O(n^3)$
- **Robustness:** Stable across diverse architectures
- **Performance:** Near-optimal convergence in practice
- **Accessibility:** Easy implementation and tuning

Takeaway: While we don't need the exact Hessian, we absolutely need the curvature intuition it provides. Modern deep learning thrives on smart approximations that capture just enough second-order information to accelerate learning while maintaining computational feasibility.