Day 25: Hessian-Vector Products: Curvature-Aware Optimization

Leveraging Second-Order Information Without Explicit Hessians

1. The Need for Second-Order Information

First-order optimization methods (like gradient descent) use only gradient information $\nabla f(x)$. However, they can be inefficient on ill-conditioned problems where the loss landscape has very different curvature in different directions.

Second-order methods use Hessian information $H(x)$ to account for curvature, but face a fundamental challenge: for a function with n parameters, the Hessian is an $n \times n$ matrix. For modern deep learning models where n can be in the millions or billions, storing H explicitly is impossible.

2. Hessian-Vector Products (HVP): The Solution

The key insight is that we rarely need the full Hessian matrix. Instead, we often need only its action on vectors through Hessian-Vector Products:

Rohit Sanwariya

$$\text{HVP}(v) = H(x)v$$

where $H(x) = \nabla^2 f(x)$ is the Hessian matrix and $v \in \mathbb{R}^n$ is an arbitrary vector.

Why HVPs Are Feasible

- The Hessian is the Jacobian of the gradient: $H(x) = J_{\nabla f}(x)$
- We can compute HVPs using the techniques we already know: JVPs and VJPs
- This avoids ever forming the explicit $n \times n$ matrix
- Computational cost is similar to evaluating $f(x)$ (typically $O(n)$)

3. Computing HVPs: Three Equivalent Approaches

There are several equivalent ways to compute HVPs, all based on automatic differentiation:

Method 1: Forward-over-Reverse Mode

$$\begin{aligned}\text{HVP}(v) &= H(x)v \\ &= J_{\nabla f}(x)v \quad \text{(JVP of the gradient)}\end{aligned}$$

This applies forward-mode autodiff to the gradient function.

Method 2: Reverse-over-Forward Mode

$$\text{HVP}(v) = \nabla(v^\top \nabla f(x))$$

This computes the gradient of the directional derivative.

Method 3: Finite Difference Approach

$$\text{HVP}(v) \approx \frac{\nabla f(x + \epsilon v) - \nabla f(x)}{\epsilon}$$

For small ϵ, this provides a numerical approximation (though less stable than autodiff methods).

Rohit Sanwariya

4. A Concrete Example: Quadratic Function

Consider the quadratic function:

$$f(x) = \frac{1}{2}x^\top Ax + b^\top x + c$$

where A is symmetric positive definite.

The gradient and Hessian are:

$$\nabla f(x) = Ax + b, \quad H = A$$

For a vector v, the HVP is simply:

$$\text{HVP}(v) = Av$$

The key insight is that we can compute Av without explicitly forming A if we have a function that computes Ax efficiently.

5. The Pearlmutter Trick: Efficient HVP Computation

For general functions, we can use what's known as the Pearlmutter trick (1994) to compute HVPs efficiently:

Pearlmutter's Algorithm

1. Compute the gradient function $g(x) = \nabla f(x)$
2. Use forward-mode autodiff to compute $J_g(x)v = H(x)v$
3. Alternatively, use: $\text{HVP}(v) = \nabla(v^\top g(x))$

Both approaches avoid explicit Hessian formation.

6. Applications in Machine Learning

Where HVPs Power Modern AI

- **Newton's Method:** $x_{k+1} = x_k - H^{-1}\nabla f(x_k)$ requires solving $Hp = -\nabla f$, which can be done iteratively using HVPs
- **Hessian-Free Optimization:** Uses conjugate gradient with HVPs to approximate Newton steps
- **Natural Gradient:** Fisher information matrix acts like a Hessian; natural gradient descent uses FIM-vector products
- **Uncertainty Quantification:** Eigenvalues of H indicate flat vs sharp minima (related to generalization)
- **Neural Network Pruning:** Second-order information helps identify unimportant weights
- **Differential Equation Solving:** HVPs appear in implicit differentiation and adjoint methods

7. Implementation in Modern Frameworks

Most deep learning frameworks provide HVP functionality:

- **PyTorch:** `torch.autograd.functional.vhp` (Vector-Hessian Product)
- **JAX:** `jax.jvp(jax.grad(f))` or `jax.grad(lambda x: jax.grad(f)(x) @ v)`

- **TensorFlow:** `tf.autodiff.ForwardAccumulator` for forward-over-reverse

8. Practical Considerations

Challenges and Solutions

- **Memory Usage:** HVPs require storing intermediate values for second derivatives
- **Numerical Stability:** Second derivatives can be numerically challenging
- **Stochastic Estimates:** For large datasets, we can use stochastic HVPs
- **Approximate Methods:** L-BFGS and other quasi-Newton methods approximate H^{-1} without explicit HVPs

9. Historical Context

- **1994:** Pearlmutter's seminal paper on efficient HVPs using automatic differentiation
- **2010:** Martens introduces Hessian-free optimization for deep learning
- **2010s:** Widespread adoption in second-order optimization methods
- **Recent:** Applications in Bayesian deep learning and uncertainty quantification

Rohit Sanwariya

Key Takeaway

Hessian-Vector Products provide access to second-order curvature information without the computational impossibility of explicit Hessian formation. By leveraging the automatic differentiation techniques we've developed (JVPs and VJPs), we can compute HVPs efficiently, enabling second-order optimization methods even for models with millions or billions of parameters. This represents the cutting edge of optimization theory meeting the practical demands of modern deep learning.