Source task and conventions

Compare forward and reverse automatic differentiation in arithmetic, storage and parallelism.

Conventional solution and proof

Consider a straight-line program with L elementary operations computing $F : \mathbb{R}^n \to \mathbb{R}^m$. A forward sweep seeded by v computes $J_F v$ with a constant amount of derivative arithmetic per node. Therefore n basis-vector sweeps recover a full Jacobian in $O(nL)$ work. A reverse sweep seeded by w computes $J_F^{\mathsf{T}} w$, so m basis-vector sweeps cost $O(mL)$. These bounds follow by applying the local chain rule once per node per seed. A straightforward reverse implementation stores $O(L)$ primal intermediates. Forward mode stores tangents with the live primal values, requiring $O(W)$ storage per seed when W is the maximum live width. Checkpointing can trade reverse storage for recomputation. Independent seeds and independent graph branches can be parallelized, but dependency chains retain their depth.

Structural or analytic derivation

Use forward mode for a small input dimension, directional derivatives or Jacobian–vector products. Use reverse mode for a scalar loss with many parameters, or vector–Jacobian products. Hessian–vector products can combine a forward directional sweep with reverse gradients. Holomorphic functions admit complex derivatives, while general real losses on complex arrays require the real differential; the mode choice does not remove this distinction.

Comparative cost and preferred method