Backpropagation with implicit layers

Naive (explicit) differentiation

- We have multiple Newton's updates
- We must apply the chain rule on the sequence of z for all time steps of updates
- Quite expensive with large number of iterations
- We must store lots of intermediate values and Jacobians in memory
- Also, chain rule in long sequences can be quite unstable (vanishing/exploding)

Implicit differentiation

- What if we could avoid iterations during backdrop altogether?
- In backprop, for layer $z = f(x)$ we are mainly interested in computing the gradient of a layer's output w.r.t. to the layer's input, that is $\dfrac{\partial z}{\partial x}$
- Let's denote the fixed point solution of $g(x, z)$ as $z^\star(x)$ with Jacobian $\dfrac{\partial z^\star(x)}{\partial x}$
- At $z^\star(x)$ the implicit layer $g(x, z)$ does not change, that is its Jacobian is zero

$$\frac{\partial g(x, z^\star(x))}{\partial x} = 0$$

Implicit differentiation

- The $g(x, z^\star)$ is a function of two variables: x and $z^\star(x)$
- Importantly, $z^\star$ depends on the first variable x w.r.t. which we differentiate
- The chain rule "splits" x and $z^\star$ in $\dfrac{\partial g(x, z^\star(x))}{\partial x}$. Then, it computes the gradient for each variable separately, while considering the other one not varying (i.e., "fixed"). Then, it sums up the two gradients

$$\frac{\partial g(x, z^\star(x))}{\partial x} = 0 \Rightarrow \underbrace{\frac{\partial g(x, z^\star)}{\partial x}}_{\text{Considering } z^\star \text{ fixed}} + \frac{\partial g(x, z^\star(x))}{\partial z^\star}\frac{\partial z^\star(x)}{\partial x} = 0$$

Implicit differentiation

- By rearranging, we have

$$\frac{\partial g(x, z^\star)}{\partial x} + \frac{\partial g(x, z^\star(x))}{\partial z^\star}\frac{\partial z^\star(x)}{\partial x} = 0 \Rightarrow$$

$$\boxed{\frac{\partial z^\star(x)}{\partial x} = -\left(\frac{\partial g(x, z^\star(x))}{\partial z^\star}\right)^{-1}\frac{\partial g(x, z^\star)}{\partial x}}$$

- In $\frac{\partial g(x, z^\star)}{\partial x}$), the chain rule considers the $z^\star(x)$ as fixed (not dependent on x), which is why we use $z^\star$ instead
- We compute $\frac{\partial g(x, z^\star)}{\partial x}$ with auto-diff at the converged $z^\star$ obtained from forward prop

Implicit function theorem

- More generally, for a continuously differentiable function f with non-singular Jacobian, and with roots at $a_0, z_0 : f(a_0, z_0) = 0$
- There is a unique continuous function $z^\star : S_{\alpha_0} \to S_{z_0}$ representing all fixed-points

$$z_0 = z^\star(\alpha_0) \quad \text{such that} \quad f(\alpha, z^\star(\alpha)) = 0 \;\; \forall \alpha \in S_{\alpha_0}$$

- where S_{α_0}, S_{z_0} open sets in the parameter and input space

Implicit differentiation: Things to consider

- In practice, we cannot compute the inverse $\left(\frac{\partial g(x, z^\star(x))}{\partial z^\star}\right)^{-1}$ directly
- An iterative process is needed instead
- Importantly, with the implicit differentiation it does not matter what algorithm/solver we use to find the root $z^\star$
- Note: $\frac{\partial g(x, z^\star)}{\partial z^\star}$ both in the forward prop of Newton's method, as well as the back prop of implicit differentiation

Implementing implicit differentiation

- Given our loss function ℓ the gradient is $\dfrac{\partial \ell}{\partial x} = \dfrac{\partial \ell}{\partial z^\star}\dfrac{\partial z^\star}{\partial x} = -\dfrac{\partial \ell}{\partial z^\star}\left(\dfrac{\partial g}{\partial z^\star}\right)^{-1}\dfrac{\partial g}{\partial x}$
- No need to compute the full Jacobian $\dfrac{\partial z^\star}{\partial x}$ or the full $\dfrac{\partial g}{\partial z^\star}$ and its inverse
- Simple vector-Jacobian product; we can use a linear equation solver

$$xA = b \Leftrightarrow x = bA^{-1}$$