JAX best practices

rough notes by skye Oct 16, 2024

- Can you jit anywhere? What's up with jit(inline=True)
    - Yes jit anywhere. Don't worry about inline=True, the end program that's generated is always fully inlined
- vmap(jit) vs. jit(vmap) (or jit(vmap(jit))
    - You get the same program out. But if you don't jit at the outermost leve, you will retrace every time (semantically equivalent but potentially slower)
- manual broadcasting vs. vmap: we ended up with identical functions in the end
    - → don't fear the vmap
- scan vs. vmap
    - scan is sequential, vmap is batched (since hardware usually supports big batched operation)
        - vmap could boil down to sequential operations if hardware doesn't support it…
    - scan can save memory over a python for loop
- How do I see how many e.g. numpyro jit'd calls are being generated?
    - jax.log_compiles()
    - jax.explain_cache_misses()
    - use the profiler if you're really fancy
- ravel_index hard to use because can't do logic on tracer objects
    - tracer shapes are just ints, can do fancy things
    - but jnp.ravel_index will still return a tracer → can't index
    - just use numpy! inside your jit'd function
    - trace logic is the same GPU vs CPU
        - can run numpy operations in GPU computation because tracing/compiling still happens on CPU
    - sidebar: looking at jaxpr of jit(lambda x: np.eye(x.shape[0]) @ x)
        - inputs at the top: two inputs!
        - jaxpr only contains dot product of those two inputs
            - numpy operations never show up in jaxpr
    - can/should we do all the indexing logic using jnp instead np? (without the tracers)
        - maybe! ends up getting inlined into program. xla will optimize, could be good, or could be worse if it's bad at optimizing
            - can check with compile().as_text()
        - doesn't always work if you're doing stuff that can't be done with jax, then just use "regular" python