What's JAX

John Stachurski

2025

Topics

- Foo
- Bar

Focus on JAX

https://jax.readthedocs.io/en/latest/

- Just-in-time compilation
- Automatic differentiation
- Xccelerated linear algebra

```python
import jax.numpy as jnp
from jax import grad, jit

def f(θ, x):
  for W, b in θ:
    w = W @ x + b
    x = jnp.tanh(w)
  return x

def loss(θ, x, y):
  return jnp.sum((y - f(θ, x))**2)

grad_loss = jit(grad(loss))  # Now use gradient descent
```

Example. AlphaFold3 (Google JAX)

Highly accurate protein structure prediction with AlphaFold

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool,...

Nature Vol. 596 (2021)

- Citation count = $30\mathrm{K}$
- Nobel Prize in Chemistry 2024

Functional Programming

JAX adopts a **functional programming style**

Key feature: Functions are pure

- Deterministic: same input $\implies$ same output
- Have no side effects (don't modify state outside their scope)

A non-pure function

```python
tax_rate = 0.1  # Global
price = 10.0    # Global

def add_tax_non_pure():
    global price
    # The next line both accesses and modifies global state
    price = price * (1 + tax_rate)
    return price
```

A pure function

```
def add_tax_non_pure(price, tax_rate=0.1):
    price = price * (1 + tax_rate)
    return price
```

General advantages:

- Helps testing: each function can operate in isolation
- Data dependencies are explicit, which helps with understanding and optimizing complex computations
- Promotes deterministic behavior and hence reproducibility
- Prevents subtle bugs that arise from mutating shared state

Advantages for JAX:

- Functional programming facilitates autodiff because pure functions are more straightforward to differentiate (don't mod external state
- Pure functions are easier to parallelize and optimize for hardware accelerators like GPUs (don't depend on shared mutable state, more independence)
- Transformations can be composed cleanly with multiple transformations yielding predictable results
- Portability across hardware: The functional approach helps JAX create code that can run efficiently across different hardware accelerators without requiring hardware-specific implementations.

JAX PyTrees

A PyTree is a concept in the JAX library that refers to a tree-like data structure built from Python containers.

Examples.

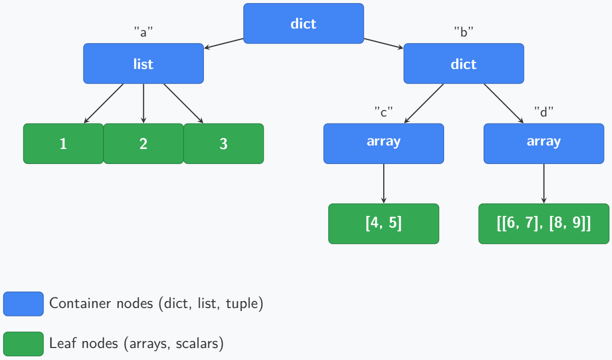
- A dictionary of lists of parameters
- A list of dictionaries of parameters, etc.

JAX can

- apply functions to all leaves in a PyTree structure
- differentiate functions with respect to the leaves of PyTrees
- etc.

JAX PyTree Structure

```
pytree = {
    "a": [1, 2, 3],
    "b": {"c": jnp.array([4, 5]), "d": jnp.array([[6, 7], [8, 9]])}
}
```



```python
# Apply gradient updates to all parameters
def sgd_update(params, grads, learning_rate):
    return jax.tree_map(
        lambda p, g: p - learning_rate * g,
        params,
        grads
    )

# Calculate gradients (PyTree with same structure as params)
grads = jax.grad(loss_fn)(params, inputs, targets)

# Update all parameters at once
updated_params = sgd_update(params, grads, 0.01)
```