Modelling Computation as Tensors

Nimalan M

Nov 23

By Nimalan M
Github - (@mark1626)

Linear Algebra and Tensors

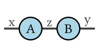
Tensors are a more nature way to representing computational problems which are derived from Linear Algebra. Modelling computations as tensors saves the scientist a lot of effort in programming and allows them to focus more on the problem they are solving.

Let us consider for this case that a tensor is a multidimensional array. A rank zero tensor is a scalar, rank-1 tensor is a vector and rank-2 tensor is a matrix. A tensor contraction is the summation over all values of indices of a set of tensors.

For example matrix multiplication can be represented in the Einstein notation or index notation as

$$C_{ij} = A_{ik}B_{kj}$$

or in the Penrose graph notation as



Computation of Tensors

A lot of libraries these days allow working with tensors. Examples in Python numpy, cupy, jax, etc and even machine learning libraries like tensorflow and pytorch. In C++ there is xtensor, MatX etc.

```
import numpy as np

a = np.randn(2, 5)
b = np.randn(5, 4)

# Matrix multiplication
c = a @ b

l = np.randn(2, 3, 4)
m = np.randn(3, 4, 5)
n = np.randn(2, 3, 5)

# Tensor reduction
np.einsum("ijk,jkx,ijx->ij", l, m, n)
```

Composing Transforms

The true benefit of modelling computations as tensors arise when we chose libraries that allow for composing transforms on tensors.

```
import jax.numpy as np
from jax import jit

@jit
def fn(B, C, D):
  return (B * cos(C)) / D

@jit
def fn2(A, B, C):
  D = jnp.einsum("ijk,ij->ij", A, C)
  E = jnp.einsum("ijk,ij->ij", B, C)
  return jnp.stack([D, E])
```

In the above example the `@jit` tag, performs a JIT compilation and composed the transformations. Depending on the nature of the library even more complex transformations can be composed.

Tensors and Category Theory

Tensors form a category. The Vec category has vector spaces as objects and linear maps between vector spaces are morphisms. As an example let $M : A \to B$, $N : B \otimes C \to D$, $P : D \to E$ be linear maps between finite vector spaces A, B, C, D, E. These can be combined to obtain $F : A \otimes C \to E$, or this can be written as

$$F = P \circ N \circ (M \otimes id_c)$$

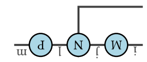
In the mathematical notation this is

$$f_{m,k} = \sum_j \sum_l p_{m,l} n_{l,jk} m_{j,i}$$

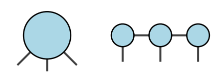
In the einstein notation this is

$$F^{ik}_m = P^l_m N^{jk}_l M^i_j$$

And in the penrose graphical notation



Tensor Decomposition



Another technique for computing tensors in hardware is to decompose higher order tensors into multiple smaller order tensors. These smaller order tensors are then batched and processed in parallel in hardware.

```
import jax.numpy as jnp

#  ([a], [a]) -> []
vv = lambda x, y: jnp.vdot(x, y)

#  ([b,a], [a]) -> [b]
mv = vmap(vv, (0, None), 0)

#  ([b,a], [a,c]) -> [b,c]
mm = vmap(mv, (None, 1), 1)
```

The above is an example of starting with a order 1 tensor operation vector dot product and deriving matrix multiplication

Hardware Acceleration

A lot of modern hardware accelerators have dedicated tensor processors/core to accelerate tensor operations.

GPUs with tensor cores usually have better performance when working with tensors.

jax uses LLVM's MLIR to target GPUs and Google TPUs.

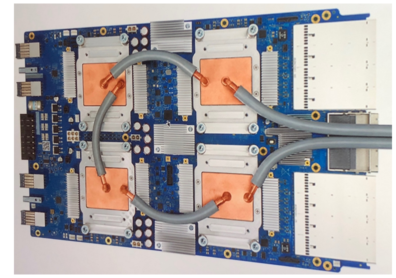


Figure 1: Google TPU 3.0