xtensor

Fast Data Structures for Data Sciences

Sylvain Corlay – SciPy 2018 Tools Plenary Session

@SylvainCorlay @QuantStack

What is xtensor?

A modern C++ template Library for fast N-D tensor algebra. TLDR: **NumPy for C++**

xtensor

Language bindings, to operate *in-place* on the data structures of the main languages of data science.

xtensor python

xtensor R

xtensor julia

Solid foundation meant for *speed* and ease of use

xsimd

xtl

But Also:

FFT, Linear Algebra bindings

xtensor fftw

xtensor Blas

Utilities to read & write various file formats from and to xtensor (npy/npz, HDF5, image files, audio files)

xtensor I/O

A C++ data frame in the making

xframe

An much more! ROS bindings, a cookiecutter project for authoring Python or Julia extensions...

From NumPy to xtensor

Cheat sheet available at http://xtensor.readthedocs.io/

Longer examples from NumPy translated in https://github.com/wolfv/numpy-benchmarks

Containers

Python 3 - NumPy	C++ 14 - xtensor
np.array([[3, 4], [5, 6]])	xt::xarray<double>({{3, 4}, {5, 6}}) xt::xtensor<double, 2>({{3, 4}, {5, 6}})
arr.reshape([3, 4])	arr.reshape({3, 4})
arr.astype(np.float64)	xt::cast<double>(arr)

Initializers

np.linspace(1.0, 10.0, 100)	xt::linspace<double>(1.0, 10.0, 100)
np.logspace(2.0, 3.0, 4)	xt::logspace<double>(2.0, 3.0, 4)
np.arange(3, 7)	xt::arange(3, 7)
np.eye(4)	xt::eye(4)
np.zeros([3, 4])	xt::zeros<double>({3, 4})
np.ones([3, 4])	xt::ones<double>({3, 4})
np.empty([3, 4])	xt::empty<double>({3, 4})
np.meshgrid(x0, x1, x2, indexing='ij')	xt::meshgrid(x0, x1, x2)

Broadcasting

Python 3 - NumPy	C++ 14 - xtensor
`a[:, np.newaxis]`	`xt::view(a, xt::all(), xt::newaxis())`
`a[:5, 1:]`	`xt::view(a, xt::range(_, 5), xt::range(1, _))`
`a[5:1:-1, :]`	`xt::view(a, xt::range(5, 1, -1), xt::all())`
`a[..., 3]`	`xt::strided_view(a, {xt::ellipsis, 3})`
`np.broadcast(a, [4, 5, 7])`	`xt::broadcast(a, {4, 5, 7})`
`np.vectorize(f)`	`xt::vectorize(f)`
`a[a > 5]`	`xt::filter(a, a > 5)`
`a[[0, 1], [0, 0]]`	`xt::index_view(a, {{0, 0}, {1, 0}})`

Random

`np.random.seed(0)`	`xt::random::seed(0)`
`np.random.randn(10, 10)`	`xt::random::randn<double>({10, 10})`
`np.random.randint(10, 10)`	`xt::random::randint<int>({10, 10})`
`np.random.rand(3, 4)`	`xt::random::rand<double>({3, 4})`
`np.random.choice(arr, 5)`	`xt::random::choice(arr, 5)`

Broadcasting

Python 3 - NumPy	C++ 14 - xtensor
a[:, np.newaxis]	xt::view(a, xt::all(), xt::newaxis())
a[:5, 1:]	xt::view(a, xt::range(_, 5), xt::range(1, _))
a[5:1:-1, :]	xt::view(a, xt::range(5, 1, -1), xt::all())
a[..., 3]	xt::strided_view(a, {xt::ellipsis, 3})
np.broadcast(a, [4, 5, 7])	xt::broadcast(a, {4, 5, 7})
np.vectorize(f)	xt::vectorize(f)
a[a > 5]	xt::filter(a, a > 5)
a[[0, 1], [0, 0]]	xt::index_view(a, {{0, 0}, {1, 0}})

Random

np.random.seed(0)	xt::random::seed(0)
np.random.randn(10, 10)	xt::random::randn<double>({10, 10})
np.random.randint(10, 10)	xt::random::randint<int>({10, 10})
np.random.rand(3, 4)	xt::random::rand<double>({3, 4})
np.random.choice(arr, 5)	xt::random::choice(arr, 5)

Stack, concatenate, split

Python 3 - NumPy	C++ 14 - xtensor
`np.stack([a, b, c], axis=1)`	`xt::stack(xtuple(a, b, c), 1)`
`np.concatenate([a, b, c], axis=1)`	`xt::concatenate(xtuple(a, b, c), 1)`
`np.squeeze(a)`	`xt::squeeze(a)`
`np.expand_dims(a, 1)`	`xt::expand_dims(a ,1)`
`np.atleast_3d(a)`	`xt::atleast_3d(a)`
`np.split(a, 4, axis=0)`	`xt::split(a, 4, 0)`

Iteration

`for x in np.nditer(a):`	`for(auto it=a.begin(); it!=a.end(); ++it)`

Iterating over a with a prescribed broadcasting shape	`a.begin({3, 4})`
	`a.end({3, 4})`

Iterating over a in a row-major fashion	`a.begin<layout_type::row_major>()` `a.begin<layout_type::row_major>()`

Iterating over a in a column-major fashion	`a.begin<layout_type::column_major>()`
	`a.end<layout_type::column_major>()`

NumPy

```
def diffusion(u, tempU, iterNum):
    ""”
    Apply Numpy matrix for the Forward-Euler Approximation
    ""”
    mu = .1
    for n in range(iterNum):
        tempU[1:-1, 1:-1] = u[1:-1, 1:-1] + mu * (
            u[2:, 1:-1] - 2 * u[1:-1, 1:-1] + u[0:-2, 1:-1] +
            u[1:-1, 2:] - 2 * u[1:-1, 1:-1] + u[1:-1, 0:-2])
        u[:, :] = tempU[:, :]
        tempU[:, :] = 0.0
```

xtensor

```
auto diffusion(tensor<float, 2>& u, tensor<float, 2>& tempU, int iterNum)
{
    // Apply Numpy matrix for the Forward-Euler Approximation
    float mu = 0.1;
    for (int n = 0; n < iterNum; ++n)
    {
        view(tempU, _r|1|-1|, _r|1|-1|) = view(u, _r|1|-1|, _r|1|-1|) + mu * (
            view(u, _r|2|_|), _r|1|-1|) - 2 * view(u, _r|1|-1|, _r|1|-1|) + view(u, _r|0|-2|, _r|1|-1|) +
            view(u, _r|1|-1|, _r|2| _|) - 2 * view(u, _r|1|-1|, _r|1|-1|) + view(u, _r|1|-1|, _r|0|-2|)
        );
        u = tempU;
        tempU.fill(0.f);
    }
}
```

Xtensor is *fast*

Xtensor is continuously benchmarked for basic operation and a collection of real-life examples taken from the Pythran benchmarks.

Xtensor makes explicit use of the SIMD instructions available on the target platform for optimal performances. The xsimd project underlying the acceleration is its own project adopted beyond xtensor, and offers a new alternative to boost.simd.

xsimd

Xtensor makes use of template expressions to implement Lazy Evaluation, unrolling most of the looks and preventing allocation of temporary variables.

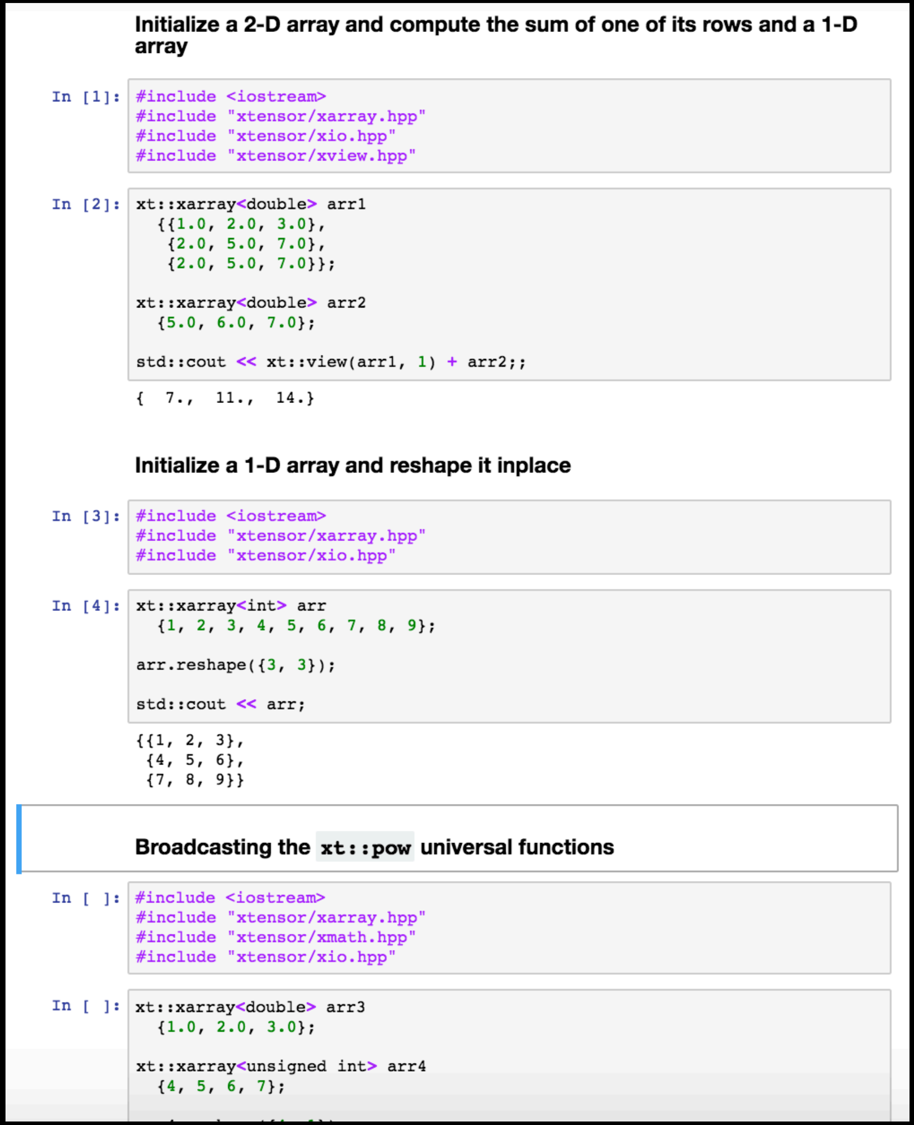
Xtensor performance

	Pairwise distance	Log likelihood	laplacian	cumsum	arc_distance	diffusion	Reverse cumsum
Python | NumPy	20020	4504	6032	3068	1080	20398	2522
xtensor	2934	2398	2210	1560	421	4949	13883
pythran	633	931	2964	1519	219	6179	2481
numba	2702	4005	8544	1476	1073	11764	1338
pypy	21991	4751	6001	2998	1150	22944	2519

A Sample from the Pythran Benchmark Suite

https://github.com/QuantStack/xtensor/

@SylvainCorlay @QuantStack



Initialize a 2-D array and compute the sum of one of its rows and a 1-D array

```
In [1]: #include <iostream>
        #include "xtensor/xarray.hpp"
        #include "xtensor/xio.hpp"
        #include "xtensor/xview.hpp"
```

```
In [2]: xt::xarray<double> arr1
          {{1.0, 2.0, 3.0},
           {2.0, 5.0, 7.0},
           {2.0, 5.0, 7.0}};

        xt::xarray<double> arr2
          {5.0, 6.0, 7.0};

        std::cout << xt::view(arr1, 1) + arr2;;
```

```
        {  7.,  11.,  14.}
```

Initialize a 1-D array and reshape it inplace

```
In [3]: #include <iostream>
        #include "xtensor/xarray.hpp"
        #include "xtensor/xio.hpp"
```

```
In [4]: xt::xarray<int> arr
          {1, 2, 3, 4, 5, 6, 7, 8, 9};

        arr.reshape({3, 3});

        std::cout << arr;
```

```
        {{1, 2, 3},
         {4, 5, 6},
         {7, 8, 9}}
```

Broadcasting the `xt::pow` universal functions

```
In [ ]: #include <iostream>
        #include "xtensor/xarray.hpp"
        #include "xtensor/xmath.hpp"
        #include "xtensor/xio.hpp"
```

```
In [ ]: xt::xarray<double> arr3
          {1.0, 2.0, 3.0};

        xt::xarray<unsigned int> arr4
          {4, 5, 6, 7};
```

Trying xtensor online

With the new C++ Kernel in a Jupyter notebook

https://mybinder.org/v2/gh/QuantStack/xtensor/0.16.4

jupyter

What is coming?

xtensor

Ongoing performance improvements:

- SIMD acceleration for complex numbers
- SIMD acceleration for `char` operations.

xframe

A C++ data frame. With a data model similar to that of pydata/xarray.

- A collection of N-D variables with labelled dimensions.
- Each variable is backed by an xtensor expression.

Just like xtensor, xframe is an ***expression system***, *lazily evaluated.*

QuantStack

Scientific Computing

Sylvain Corlay

Wolf Vollprecht

Johan Mabille

Resource

CORE:
https://github.com/QuantStack/xtl/
https://github.com/QuantStack/xsimd/
https://github.com/QuantStack/xtensor/

LANGUAGE BINDINGS
https://github.com/QuantStack/xtensor-python/
https://github.com/QuantStack/xtensor-r/
https://github.com/QuantStack/xtensor-julia/

EXTENSIONS
https://github.com/QuantStack/xtensor-io/
https://github.com/QuantStack/xtensor-blas/
https://github.com/egpbos/xtensor-fftw/
https://github.com/wolfv/xtensor_ros/

XFRAME
https://github.com/xframe

CORE:
https://xtl.readthedocs.io
https://xsimd.readthedocs.io
https://xtensor.readthedocs.io

LANGUAGE BINDINGS
https://xtensor-python.readthedocs.io
https://xtensor-r.readthedocs.io
https://xtensor-julia.readthedocs.io

EXTENSIONS
https://xtensor-io.readthedocs.io
https://xtensor-blas.readthedocs.io