RoughPy

For when your paths aren’t smooth

Sam Morley
University of Oxford

DataSig

A rough path between
mathematics and data science

University of Oxford · The Alan Turing Institute · Imperial College London · UCL · UKRI Engineering and Physical Sciences Research Council

Python + Rough Paths = RoughPy

RoughPy is a toolkit for working with streaming data through the lens of rough paths.

Design goals

1. Provide a class that represents streaming data as a rough path - we call it a stream.
   - Query over intervals to get a signature.
   - Intelligently cache intermediate results.
   - Provide an abstraction over the underlying data.
2. Provide classes for the algebraic objects: free tensors, shuffle tensors, and objects from the free Lie algebra
3. Provide utilities for working with intervals and other useful tools.
4. Should provide easy interoperability with standard libraries.

An Example

First, import RoughPy:

```
>>> import roughpy as rp
```

Make some data - converted the word “stream” into a stream of letters:

```
>>> import numpy as np
>>> data = np.array([
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],
    dtype=’int8’)
```

Get an algebra context

- Width 26 - one for each letter a-z
- Depth 2
- Rational coefficients for infinite precision

```
>>> ctx = rp.get_context(width=26,
                         depth=2,
                         coeffs=rp.Rational)
```

Construct the stream

Construct the stream using the `LieIncrementStream.from_increments` constructor:

```
>>> stream = rp.LieIncrementStream.
                from_increments(data, ctx=ctx)
```

Compute the signature of the whole stream:

```
>>> sig = stream.signature()
>>> sig
FreeTensor(width=26, depth=2, ctype=Rational)
```

```
>>> print(sig)
{ 1() 1(1) 1(5) 1(13) 1(18) 1(19) 1(20) 1/2(1,1)
  1(1,13) 1(5,1) 1/2(5,5) 1(5,13) 1/2(13,13)
  1(18,1) 1(18,5) 1(18,13) 1/2(18,18) 1(19,1)
  1(19,5) 1(19,13) 1(19,18) 1/2(19,19) 1(19,20)
  1(20,1) 1(20,5) 1(20,13) 1(20,18) 1/2(20,20) }
```

Or the log signature

```
>>> print(stream.log_signature())
{ 1(1) 1(5) 1(13) 1(18) 1(19) 1(20) -1/2([1,5])
  1/2([1,13]) -1/2([1,18]) -1/2([1,19]) -1/2([1,20])
  1/2([5,13]) -1/2([5,18]) -1/2([5,19]) -1/2([5,20])
  -1/2([13,18]) -1/2([13,19]) -1/2([13,20])
  -1/2([18,19]) -1/2([18,20]) 1/2([19,20]) }
```

Or over a subinterval

```
>>> interval = rp.RealInterval(0, 3.1)
>>> print(stream.log_signature(interval))
{ 1(18) 1(19) 1(20) -1/2([18,19])
  -1/2([18,20]) 1/2([19,20]) }
```

Where we are

- Most of the algebraic types are fully implemented.
- Several increment stream types are implemented.
- Intervals and some support tools are implemented.
- Framework for device computation is in place.

Some trouble spots remain:

- Proper handling for value streams.
- Some very useful tools are missing.
- Linear operators need to be added.