Owl Summary

Highlights

- A new, self described *Numpy styled* numerical library for OCaml
- Designed to be interactive and repl friendly
- Backed by gsl and plplot

Owl Summary

Highlights

- A new, self described *Numpy styled* numerical library for OCaml
- Designed to be interactive and repl friendly
- Backed by gsl and plplot

TLDR;

- Analyze data with sane interfaces
- Make pretty plots with little hassle

About the author - Liang Wang

Credentials

- M.Sc and Ph.D degrees from University of Helsinki, Finland
- Research Associate at Computer Laboratory, Cambridge University
- Postdoc Research Associate in Queens' College

About the author - Liang Wang

Credentials

- M.Sc and Ph.D degrees from University of Helsinki, Finland
- Research Associate at Computer Laboratory, Cambridge University
- Postdoc Research Associate in Queens' College

In his own words

My work focuses on information-centric networking, network architecture, network optimisation and protocol design. I have strong interest in data analytics and big data framework. I also teach, I am an Associated Fellow in the British Higher Education Academy.

Owl vs Gsl

Differences

- developed over a decade ago by Markus Mottl (NYC)
- bindings to Gnu Scientific Library (C, but portable)
- makes no attempt to simplify somewhat opaque GSL interfaces
- sometimes used as middleware for other math libs (owl, pareto, gpr, libsvm)

Owl vs Oml

(Besides the upside-down w)

Differences

- developed mostly by Leonid Rozenberg of Hammerlab (NYC) in 2015-16
- does not depend on gsl (instead- lacaml,lbfgs,ocepheps)
- no built in plotting
- provides general math utils but centers around machine learning
- usable in repl but not (yet) as convenient for typical data

Owl vs Pareto

Differences

- Authored by Sergei Lebedev in 2013
- Depends on gsl
- Focuses primarily on statistics
- Provides common statistical tests for significant differences between samples.
- Makes uniform interfaces for common discrete and continuous probability distributions.
- Features sample statistics, quantile estimation, kernel density estimation.

Ocaml And Data Science

Leading data science languages

- Python - untyped by design
- R - awkward type system, not a general purpose language
- Matlab - proprietary, types weak or nonexistant
- Java and C# - verbose, type systems inferior to ML / Haskell
- Scala - still too verbose, suffers from some Java hangups
- F# - an interesting choice, if you can tolerate MS ecosystem
- Julia - gaining momentum, something to watch out for

Ocaml And Data Science

Leading data science languages

- Python - untyped by design
- R - awkward type system, not a general purpose language
- Matlab - proprietary, types weak or nonexistant
- Java and C# - verbose, type systems inferior to ML / Haskell
- Scala - still too verbose, suffers from some Java hangups
- F# - an interesting choice, if you can tolerate MS ecosystem
- Julia - gaining momentum, something to watch out for

Why not...

- **OCaml - perfect choice but needs more data science tooling!**

Ocaml And Data Science

Leading data science languages

- Python - untyped by design
- R - awkward type system, not a general purpose language
- Matlab - proprietary, types weak or nonexistant
- Java and C# - verbose, type systems inferior to ML / Haskell
- Scala - still too verbose, suffers from some Java hangups
- F# - an interesting choice, if you can tolerate MS ecosystem
- Julia - gaining momentum, something to watch out for

Why not...

- **OCaml - perfect choice but needs more data science tooling!**
- **OCaml should be gunning for data scientists!**

A quick example

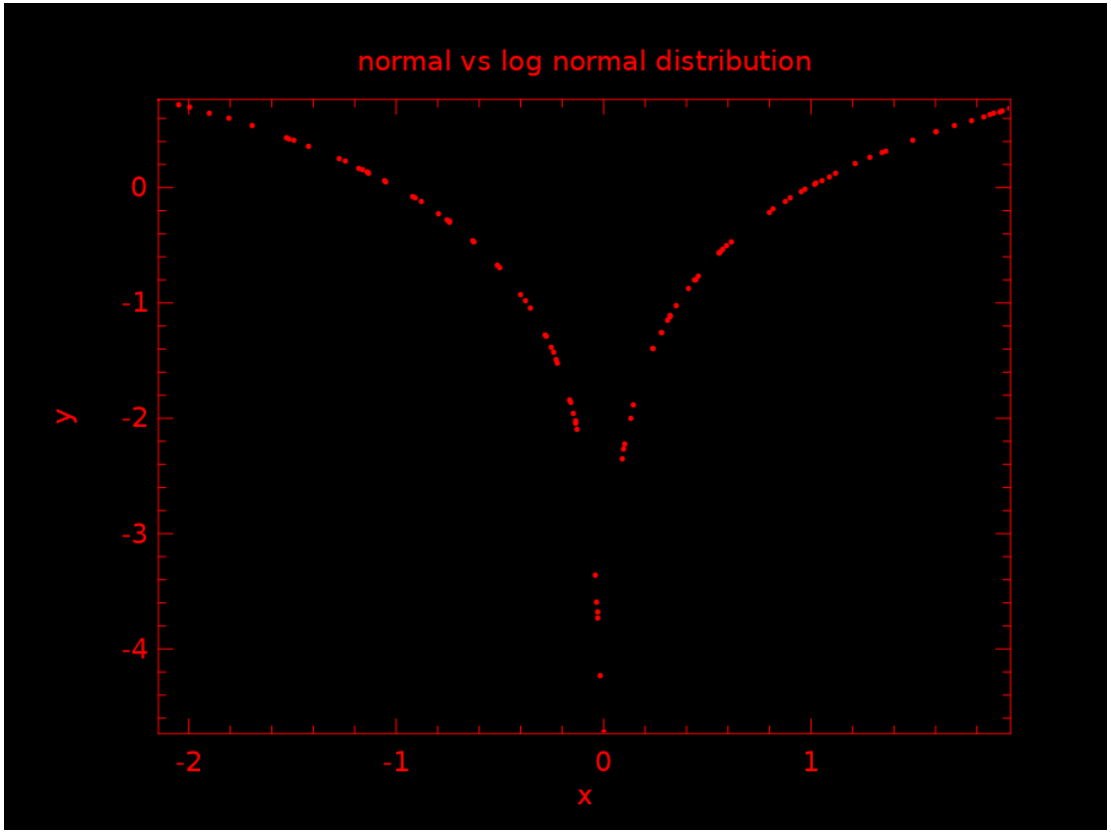
```
open Owl
module M = Dense.Real

let () =
let h = Plot.create "norm-vs-log.png" in
Plot.set_title h "normal vs log normal distribution";
let x = M.gaussian 100 1 in
let y = M.(abs x |> log) in
Plot.scatter ~h x y;
Plot.output h
```

A quick example

```
open Owl
module M = Dense.Real

let () =
let h = Plot.create "norm-vs-log.png" in
Plot.set_title h "normal vs log normal distribution";
let x = M.gaussian 100 1 in
let y = M.(abs x |> log) in
Plot.scatter ~h x y;
Plot.output h
```



Modules (Types, Matrices, and Utils)

- **Owl_types** - Define the types shared by various modules
- **Owl_utils** - Helper functions used in the library
- **Owl_const** - This module is extended from gsl-ocaml by including SI system.
- **Owl_dense** - Dense matrix module
- **Owl_dense_complex** - Complex dense matrix module: this module supports operations on dense matrices of complex numbers.
- **Owl_dense_real** - Real dense matrix module
- **Owl_sparse** - Sparse matrix module
- **Owl_sparse_complex** - Complex sparse matrix: this module supports the operations on sparse matrices of complex numbers.
- **Owl_sparse_real** - Sparse real matrix: this module supports the operations on sparse matrices of real numbers.

Modules (Functions and plotting)

- **Owl_fft** - Fast Fourier Transforms (FFTs) : this module implements some basic FFT operations.
- **Owl_linalg** - Linear Algebra module
- **Owl_maths** - Mathematics module
- **Owl_optimise** - Machine learning library Note: C layout row-major matrix
- **Owl_plot** - Plot module The input to a plot function is supposed to be a row-based matrix.
- **Owl_pretty** - Generic matrix printing functions.
- **Owl_regression** - Regression module
- **Owl_stats** - Statistics module
- **Owl_toplevel** - Install Toplevel Printers

Live plotting in utop

Now let's try to use this thing in real time...

Acknowledgements and References

Special thanks

- Jane Street - for the space and OCaml community support!
- Liang Wang - for writing Owl
- Markus Mottl - for writing gsl

References

- Owl on github: https://github.com/ryanrhymes/owl
- Owl plotting tutorial: https://github.com/ryanrhymes/owl/wiki/Tutorial:-How-to-Plot-in-Owl?
- Owl API: http://www.cl.cam.ac.uk/ lw525/owl/
- gsl-ocaml on github: https://github.com/mmottl/gsl-ocaml
- Oml on github: https://github.com/hammerlab/oml
- Pareto on github: https://github.com/superbobry/pareto