Neural Ordinary Differential Equations with Julia

Nicolas Holland

August 25, 2020

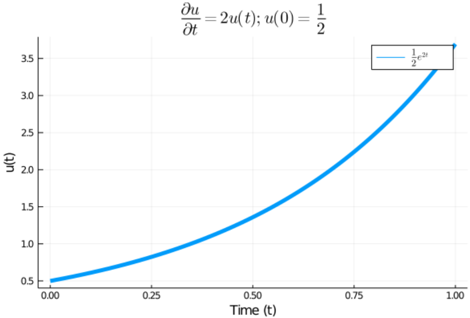
Overview

1. Theory / Math
   1. ODEs
   2. Neural Networks
   3. Neural ODEs
2. Julia Packages and Dataset
3. Neural ODE Code and Results

Let $u(t)$ be function in time, f function describing u's derivative $\frac{\partial u}{\partial t}$, starting at time t_0:

$$\frac{\partial u}{\partial t} = f(u(t), t)$$
$$u(t_0) = u_0$$

with u_0 being an initial value.



Let $u(t)$ be function in time, f function describing u's derivative $\frac{\partial u}{\partial t}$, starting at time t_0:

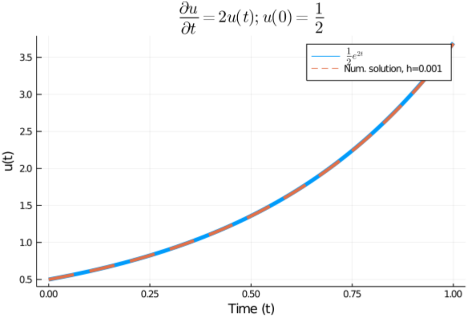
$$\frac{\partial u}{\partial t} = f(u(t), t)$$
$$u(t_0) = u_0$$

with u_0 being an initial value.

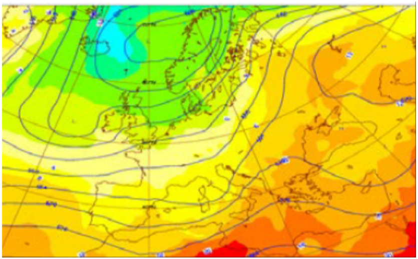
Simulate eg. using Euler method:

$$\widetilde{u}_{t+1} = u_t + hf(u_t, t)$$

with h being a step size.



Numerical Weather Prediction (NWP)



1. PDE (space and time)
2. Modelling of fluid dynamics
3. Navier-Stokes equations
4. Computational heavy

Observations

$$u_t, u_{t+1}, u_{t+2}, \cdots$$

Observations

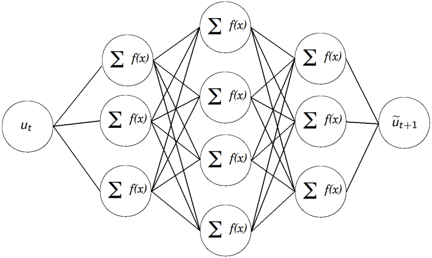
$$u_t, u_{t+1}, u_{t+2}, \cdots$$

Can we learn this?

$$f(u_t, \theta) = u_{t+1}$$

Neural Networks

Let f be a neural network with θ being its parameters.

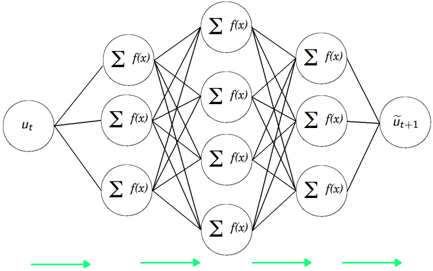


Neural Networks

Let f be a neural network with θ being its parameters.

Forward pass:

$$\widetilde{u}_{t+1} = f(u_t, \theta)$$



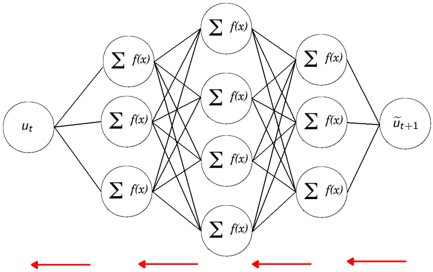
Let f be a neural network with θ being its parameters.

Forward pass:

$$\widetilde{u}_{t+1} = f(u_t, \theta)$$

Backward pass:

$$\min_{\theta} \sum_t \|\widetilde{u}_t - u_t\|$$



Neural ODEs

Instead of learning

$$\widetilde{u}_{t+1} = f(u_t, \theta)$$

we define an ODE

$$\frac{\partial u}{\partial t} = f(u(t), t)$$

and learn the change of u:

$$\frac{\partial u}{\partial t} = f(u_t, \theta)$$

Neural ODEs

Neural ODE:

$$\frac{\partial u}{\partial t} = f(u_t, \theta)$$

Forward pass:

$$\widetilde{u}_{t+1} = u_t + h \quad \underbrace{f(u_t, \theta)}_{\text{NNs forward pass}}$$

Neural ODEs

Neural ODE:

$$\frac{\partial u}{\partial t} = f(u_t, \theta)$$

Forward pass:

$$\widetilde{u}_{t+1} = u_t + h \quad \underbrace{f(u_t, \theta)}_{\text{NNs forward pass}}$$

Backward pass:

$$\min_{\theta} \sum_{t} \left\| \widetilde{u}_{t+1} - u_{t+1} \right\|$$

Neural ODEs

Neural ODE:

$$\frac{\partial u}{\partial t} = f(u_t, \theta)$$

Forward pass:

$$\widetilde{u}_{t+1} = u_t + hf(u_t, \theta)$$

Backward pass:

$$\min_{\theta} \sum_{t} \|u_t + hf(u_t, \theta) - u_{t+1}\|$$

Neural ODEs

Neural ODE:

$$\frac{\partial u}{\partial t} = f(u_t, \theta)$$

Forward pass:

$$\widetilde{u}_{t+1} = u_t + hf(u_t, \theta)$$

Backward pass (two steps):

$$\min_{\theta} \sum_t \| u_t + hf(u_t, \theta) + hf(u_t + hf(u_t, \theta), \theta) - u_{t+2} \|$$

Overview

1. ~~Theory / Math~~
   1. ~~ODEs~~
   2. ~~Neural Networks~~
   3. ~~Neural ODEs~~
2. Julia Packages and Dataset
3. Neural ODE Code and Results

Julia Packages

Flux.jl

- Machine learning in Julia
- Neural Networks, MLP, DL, ...
- Derivative computation
- GPU support
- ...

Julia Packages

Flux.jl

- Machine learning in Julia
- Neural Networks, MLP, DL, ...
- Derivative computation
- GPU support
- ...

DifferentialEquations.jl

- Define diff. equations
- Equation solvers
- ODEs, SODEs, PDEs, ...
- GPU support
- ...

Julia Packages

Flux.jl

- Machine learning in Julia
- Neural Networks, MLP, DL, ...
- Derivative computation
- GPU support
- and more!

DifferentialEquations.jl

- Define diff. equations
- Equation solvers
- ODEs, SODEs, PDEs, ...
- GPU support
- and more!

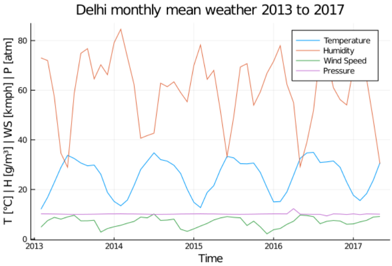
DiffEqFlux.jl

- Combination of the two
- By authors of the two
- Implementation of [1]
- By authors of [1]

[1] Chen, Ricky TQ, et al. "Neural ordinary differential equations." Advances in neural information processing systems. 2018.

Dataset

Daily climate data in the city of Delhi from 2013 to 2017. Includes mean temperature, humidity, wind speed, mean air pressure.



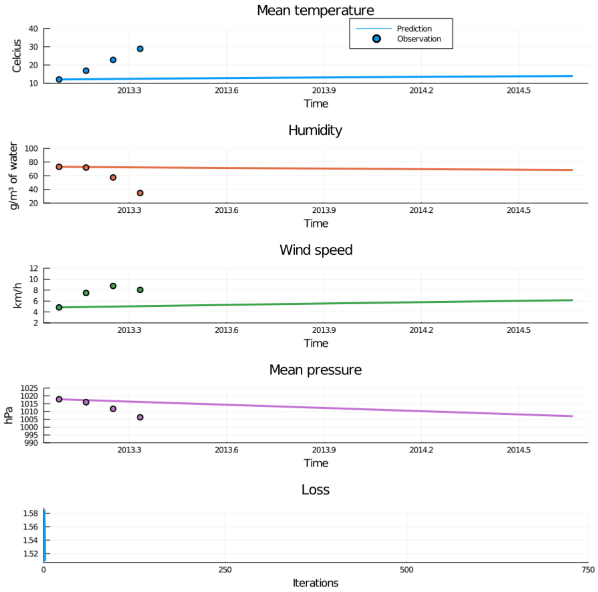
Julia Code

```julia
using DiffEqFlux

function neural_ode(t, data_dim; saveat = t)
    f = FastChain(FastDense(data_dim, 64, swish),
         FastDense(64, 32, swish),
         FastDense(32, data_dim))

    node = NeuralODE(f, (minimum(t), maximum(t)), Tsit5(),
             saveat = saveat, abstol = 1e-9,
             reltol = 1e-9)
end
```

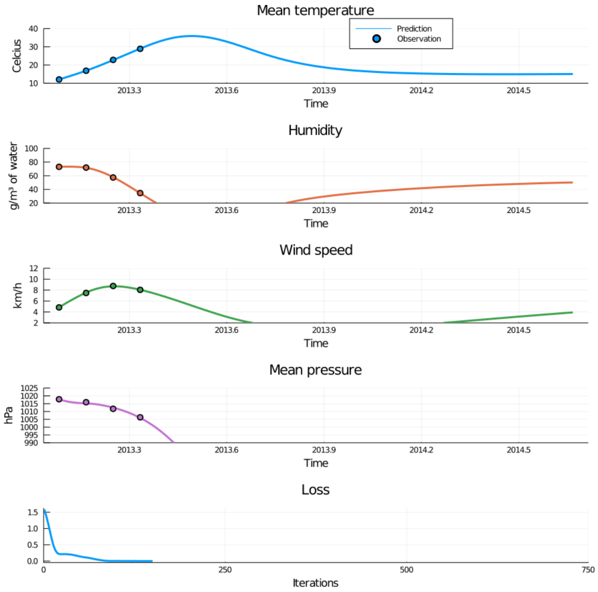
Julia Code

```julia
using OrdinaryDiffEq, Flux

function train_one_round(node, θ, u, opt, maxiters,
                         u0 = data[:, 1])
    predict(θ) = Array(node(u0, θ))
    loss(θ) = begin
        û = predict(θ)
        Flux.mse(û, u)
    end

    θ = θ == nothing ? node.p : θ
    res = DiffEqFlux.sciml_train(
        loss, θ, opt,
        maxiters = maxiters
    )
    return res.minimizer
end

node = neural_ode(train_t, 4)

θ = train_one_round(node, θ, train_u, ADAMW(1e-2), 150)
```

Results

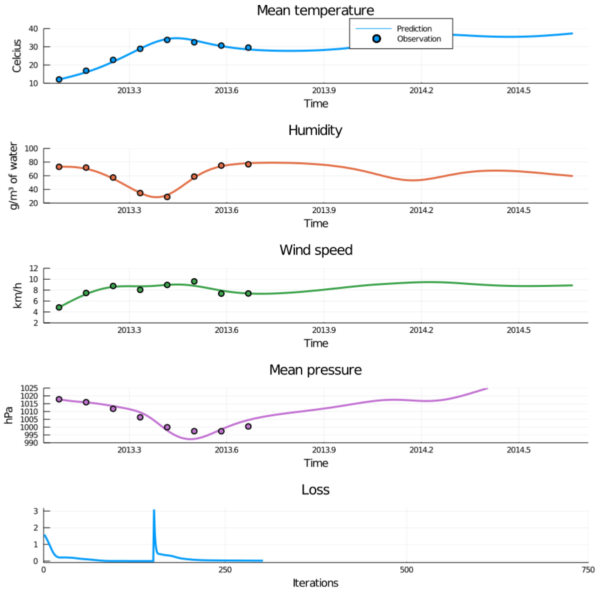

Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
1.58
1.56
1.54
1.52
0
250
500
750
Iterations

Results

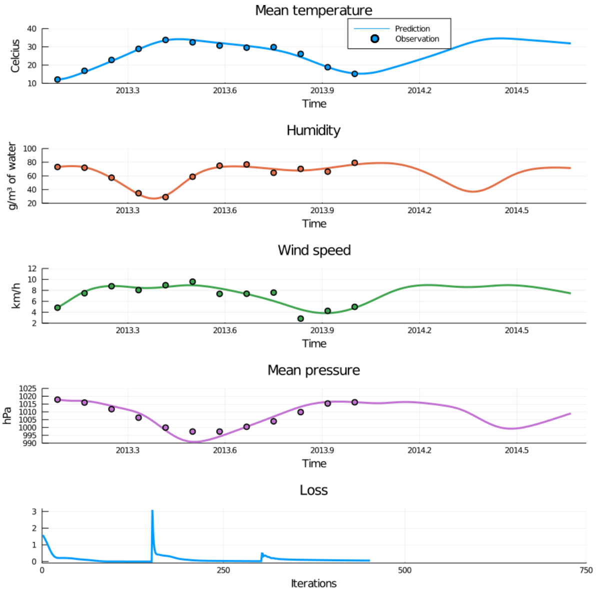


Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
1.5
1.0
0.5
0.0
0
250
500
750
Iterations

Results

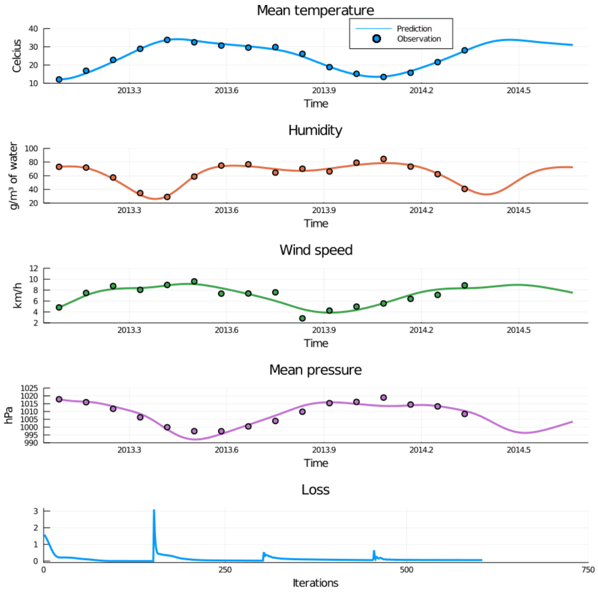

Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
3
2
1
0
0
250
500
750
Iterations

Results

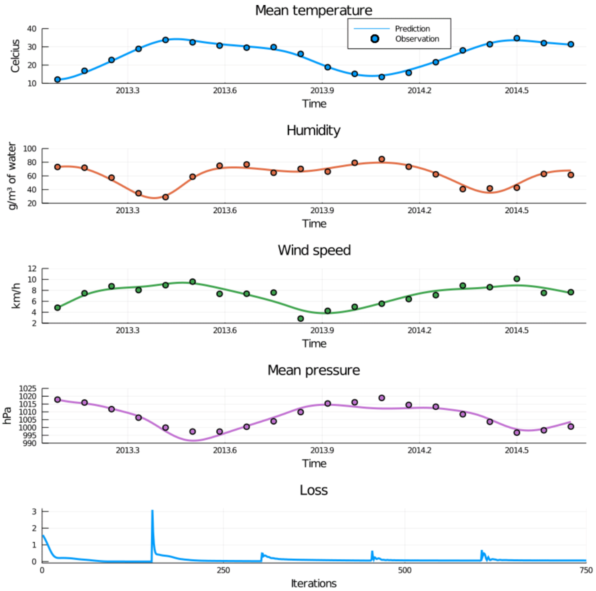


Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
3
2
1
0
0
250
500
750
Iterations

Results

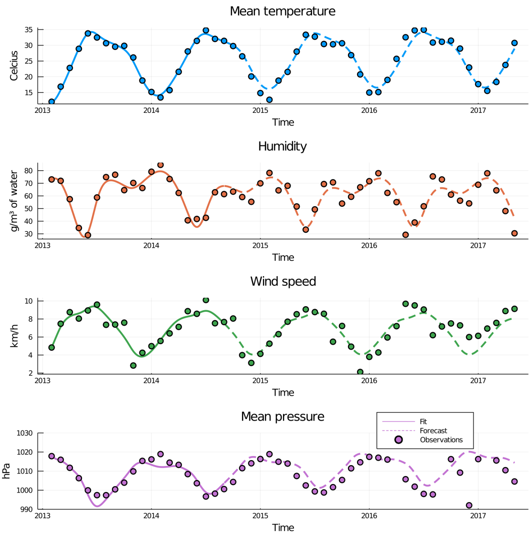

Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
3
2
1
0
0
250
500
750
Iterations

Results



Mean temperature
Prediction
Observation
Celcius
40
30
20
10
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Humidity
g/m³ of water
100
80
60
40
20
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Wind speed
km/h
12
10
8
6
4
2
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Mean pressure
hPa
1025
1020
1015
1010
1005
1000
995
990
2013.3
2013.6
2013.9
2014.2
2014.5
Time
Loss
3
2
1
0
0
250
500
750
Iterations

Results



Mean temperature
Celcius
35
30
25
20
15
2013
2014
2015
2016
2017
Time
Humidity
g/m³ of water
80
70
60
50
40
30
2013
2014
2015
2016
2017
Time
Wind speed
km/h
10
8
6
4
2
2013
2014
2015
2016
2017
Time
Mean pressure
hPa
1030
1020
1010
1000
990
2013
2014
2015
2016
2017
Time
Fit
Forecast
Observations

References

1. Chen, Ricky TQ, et al. "Neural ordinary differential equations." Advances in neural information processing systems. 2018.
2. DiffEqFlux Package
   https://github.com/SciML/DifferentialEquations.jl
3. DiffEqFlux Paper
   Chris Rackauckas et al. "DiffEqFlux.jl - A Julia Library for Neural Differential Equations", arXiv:1902.02376
4. Original Blog post this talk is based on
   sebastiancallh.github.io/post/neural-ode-weather-forecast/
5. Link to talk + my code
   github.com/nicolasholland/VariousProjects/tree/master/Neural ODE Talk
6. kaggel dataset
   www.kaggle.com/sumanthvrao/daily-climate-time-series-data