ApplicationDrivenLearning.jl

A high-performance Julia package for training predictive models in the context of decision making

Giovanni Almeida Argento de Amorim
Alexandre Street de Aguiar (PUC-Rio)
Joaquim Masset Lacombe Dias Garcia (Soma Energy)

Edinburgh, June 5, 2026

Motivation: Decision making under uncertainty

- First-level decision
  - To buy or not to buy a stock in the financial market?
  - To dispatch or not to dispatch thermal power plants to save water?
- Uncertainty is observed
  - Prices rise / fall
  - Rainfall above / below expectation
- Second-level decision and result assessment
  - Move the portfolio
  - Turn off thermal power plants, dispatch renewables
  - Suffer the consequences

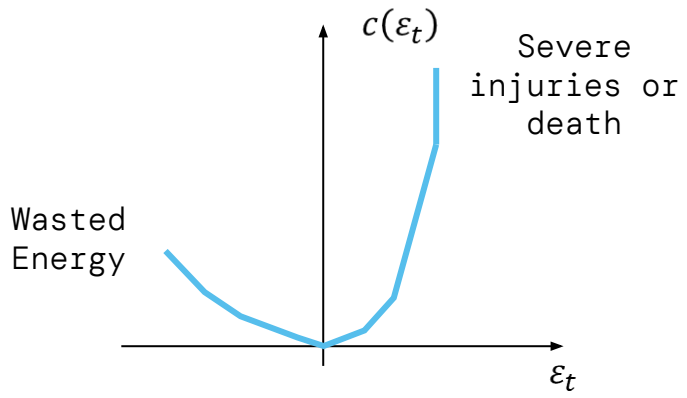
Motivation: Decision making under uncertainty

- First-level decision
  - To buy or not to buy a stock in the financial market?
  - To dispatch or not to dispatch thermal power plants to save water?
- Uncertainty is observed
  - Prices rise / fall
  - Rainfall above / below expectation
- Second-level decision and result assessment
  - Move the portfolio
  - Turn off thermal power plants, dispatch renewables
  - Suffer the consequences

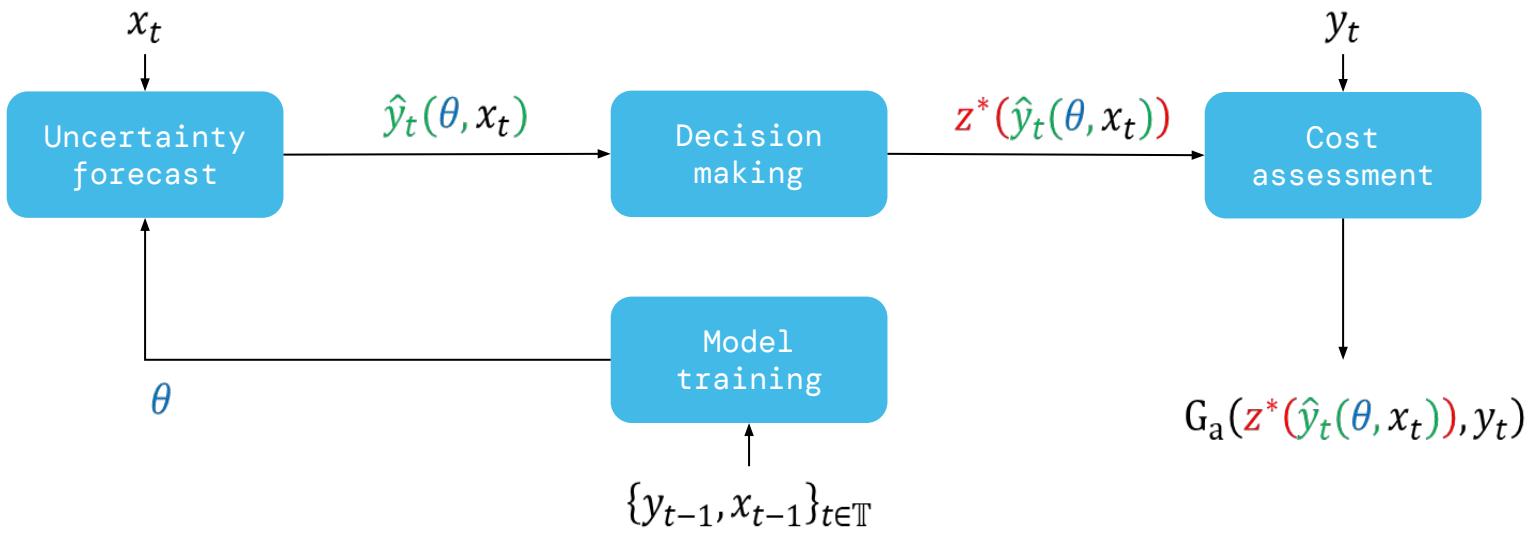
Commonly informed by uncertainty forecasts

Motivation: Assymmetric costs

Parkour: future estimates must be biased

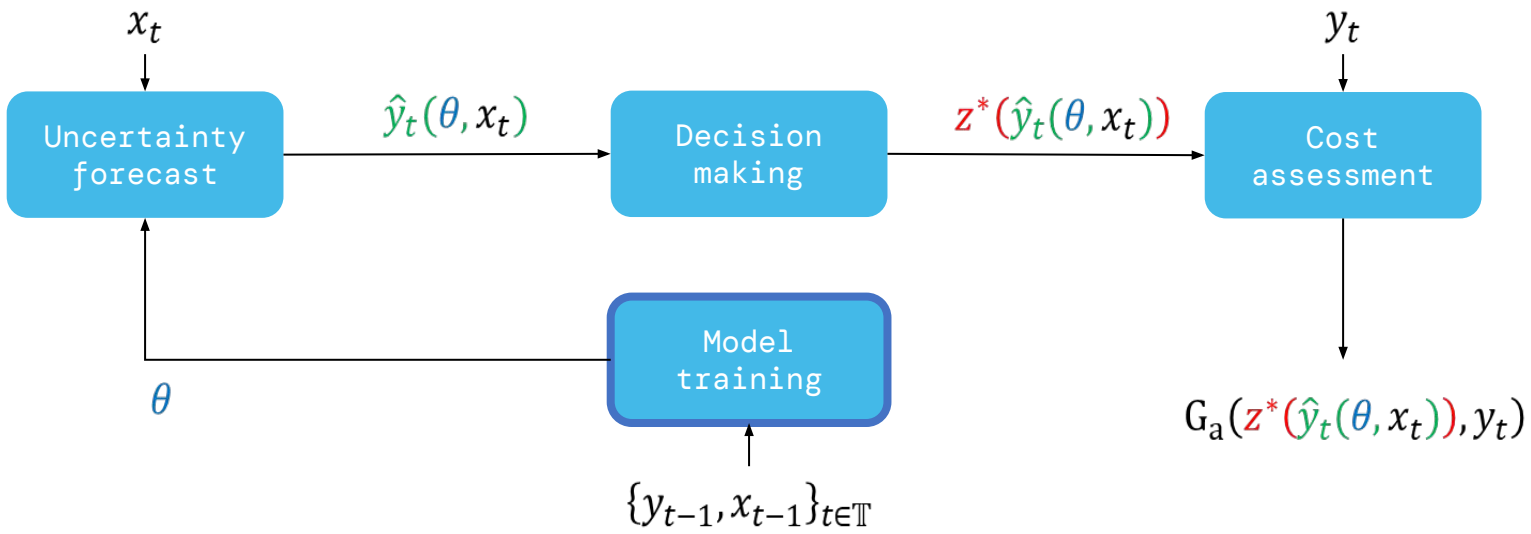


The open loop



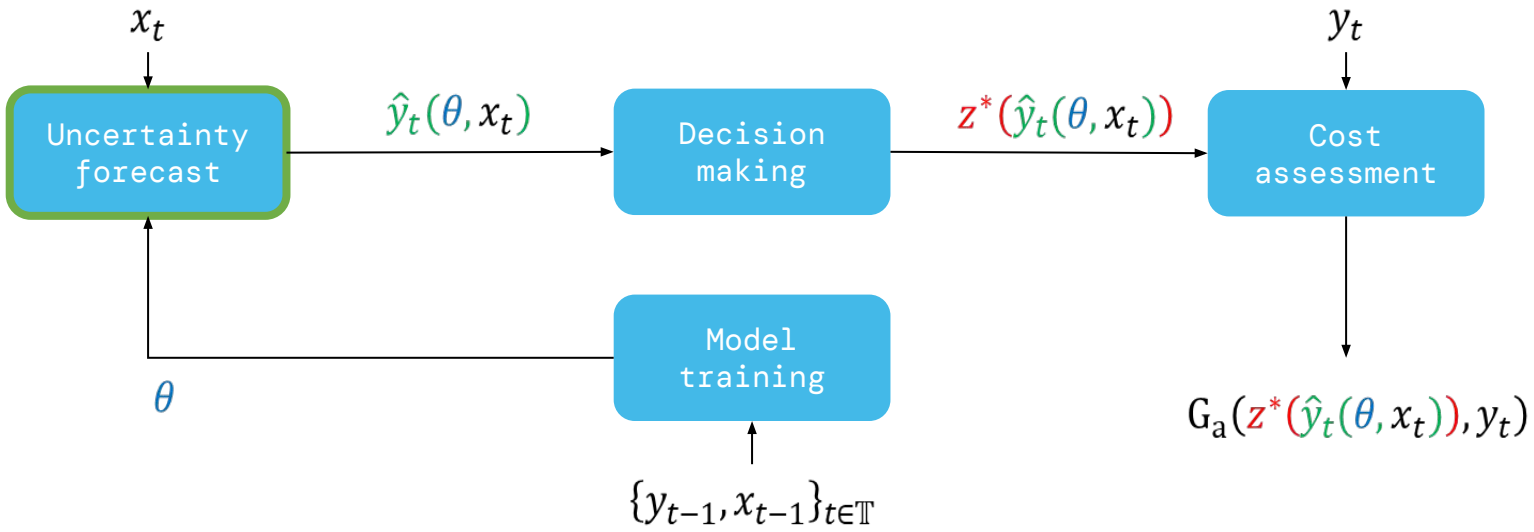
Predictive model training
Load and reserve requirements forecast
Operation Planning
Re-dispatch of resources in real time

The open loop



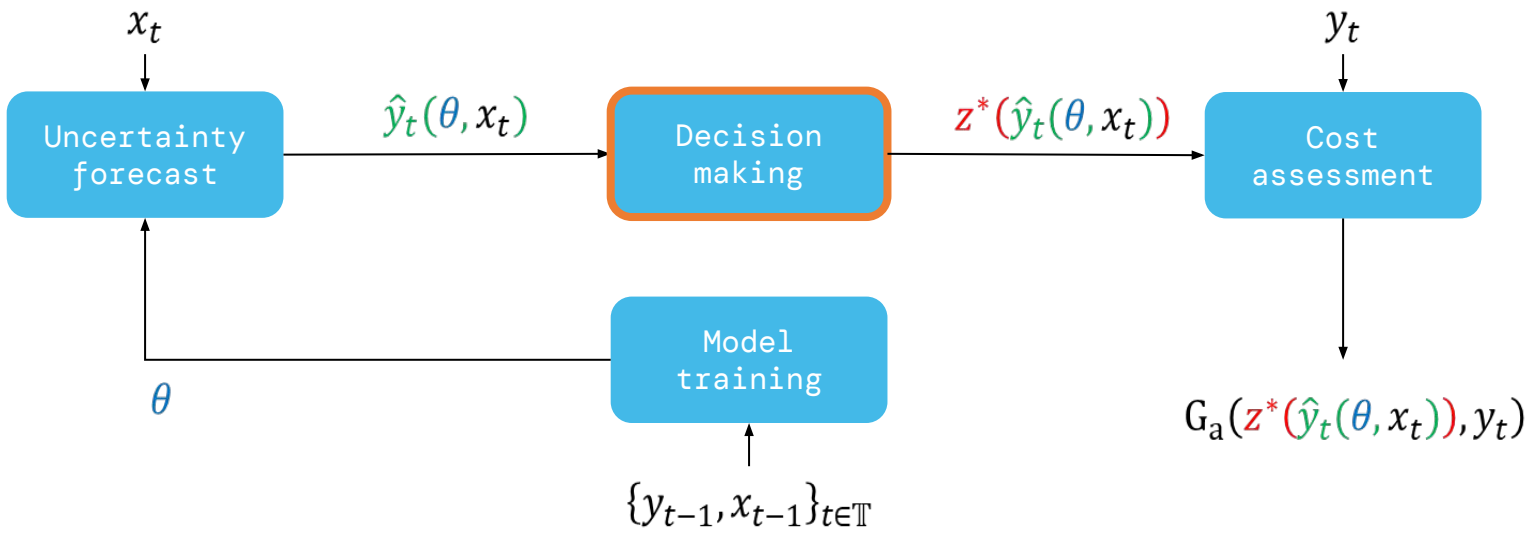
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

The open loop



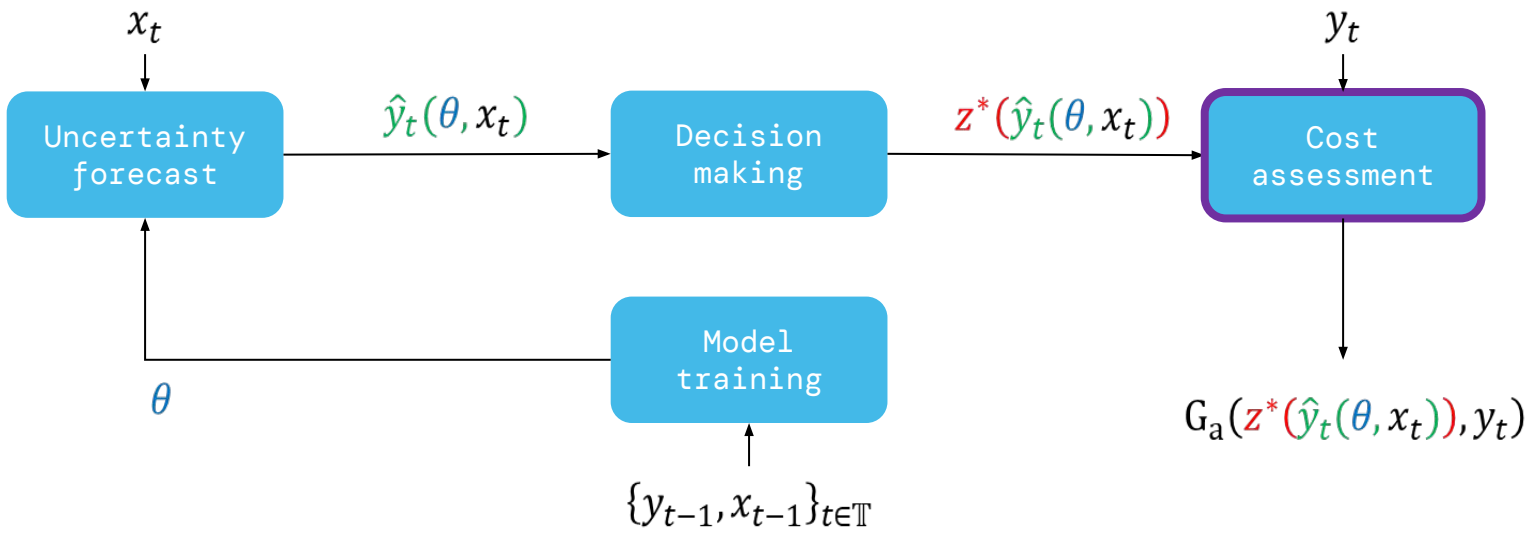
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

The open loop



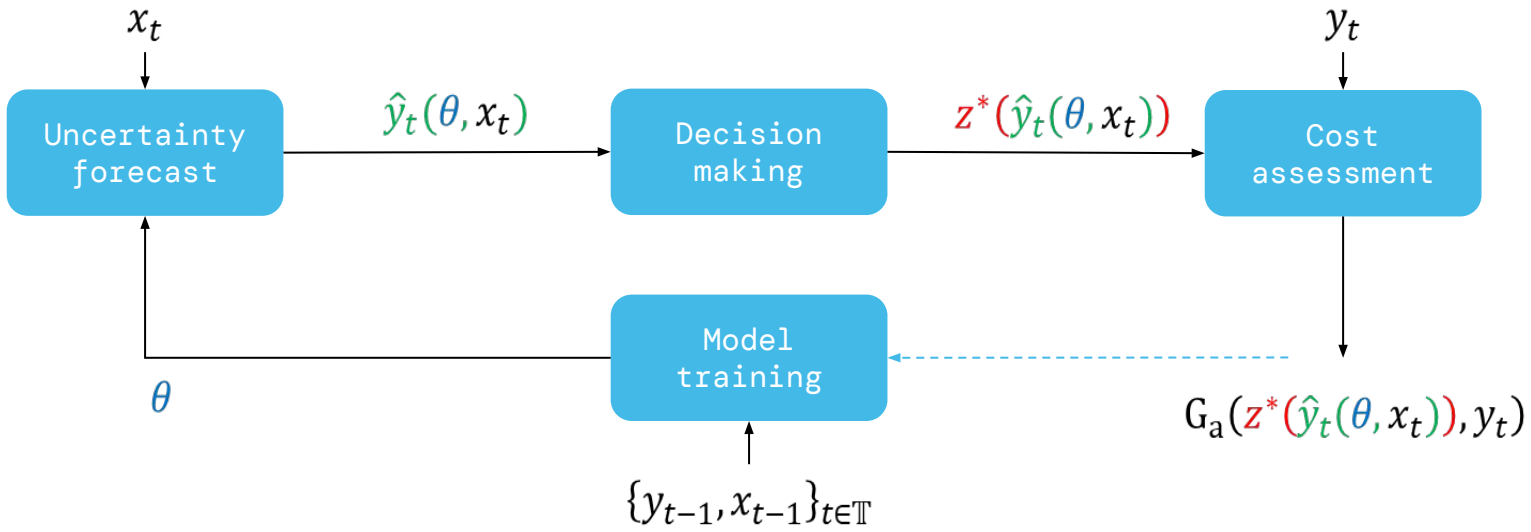
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

The open loop



- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

Closing the loop



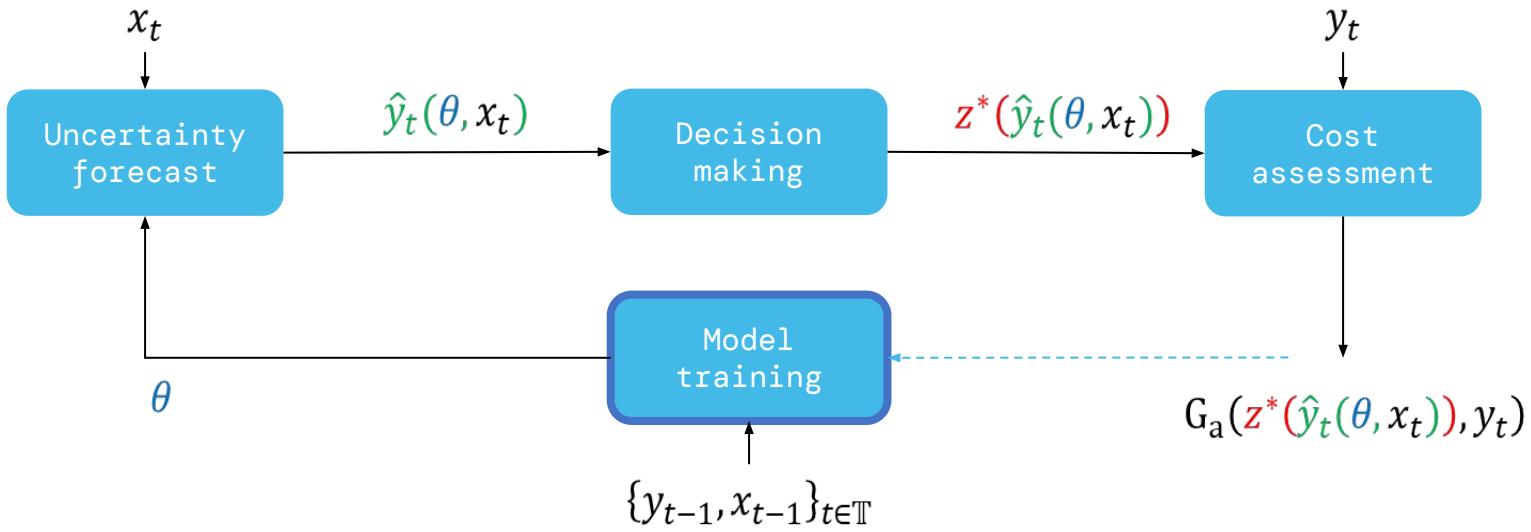
Predictive model training
Load and reserve requirements forecast
Operation Planning
Re-dispatch of resources in real time

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

Closing the loop



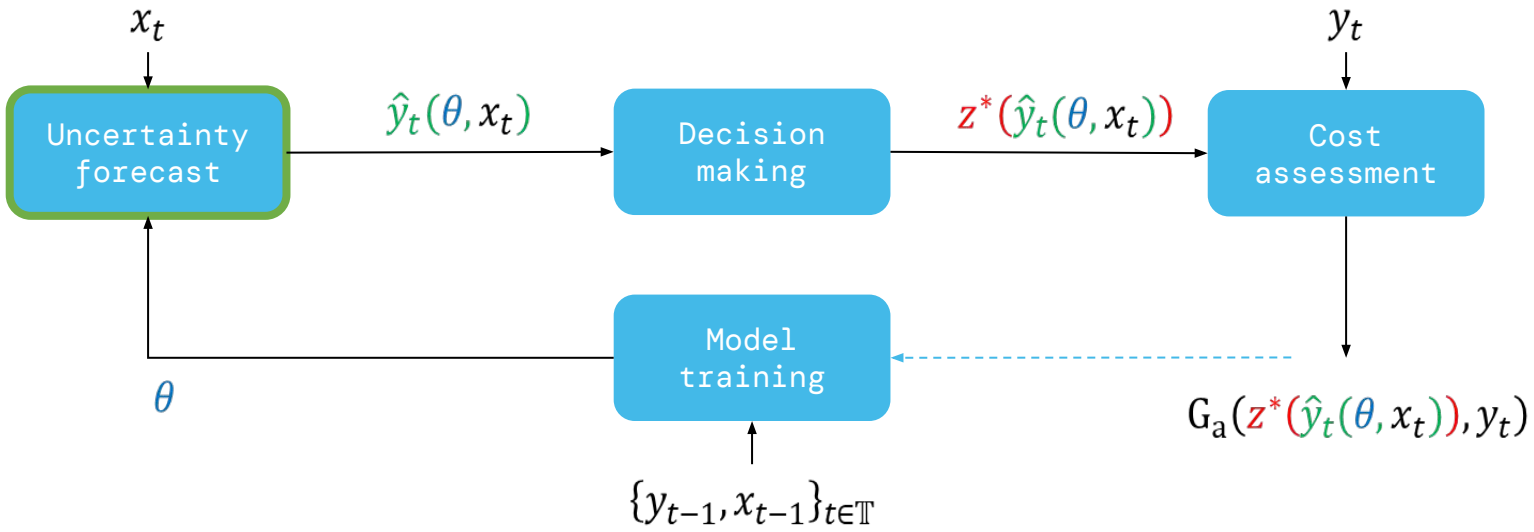
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

$$\theta_T \in \underset{\theta\in\Theta, \hat{y}_t, z_t^*}{\arg\min} \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z\in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

Closing the loop



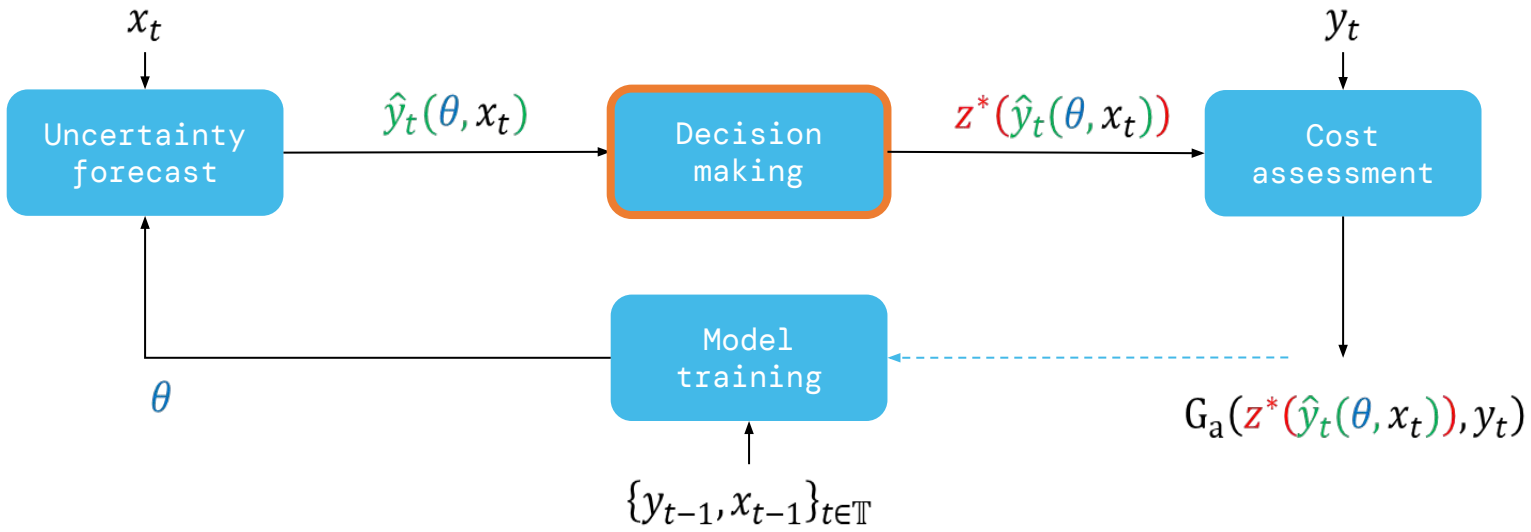
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

Closing the loop



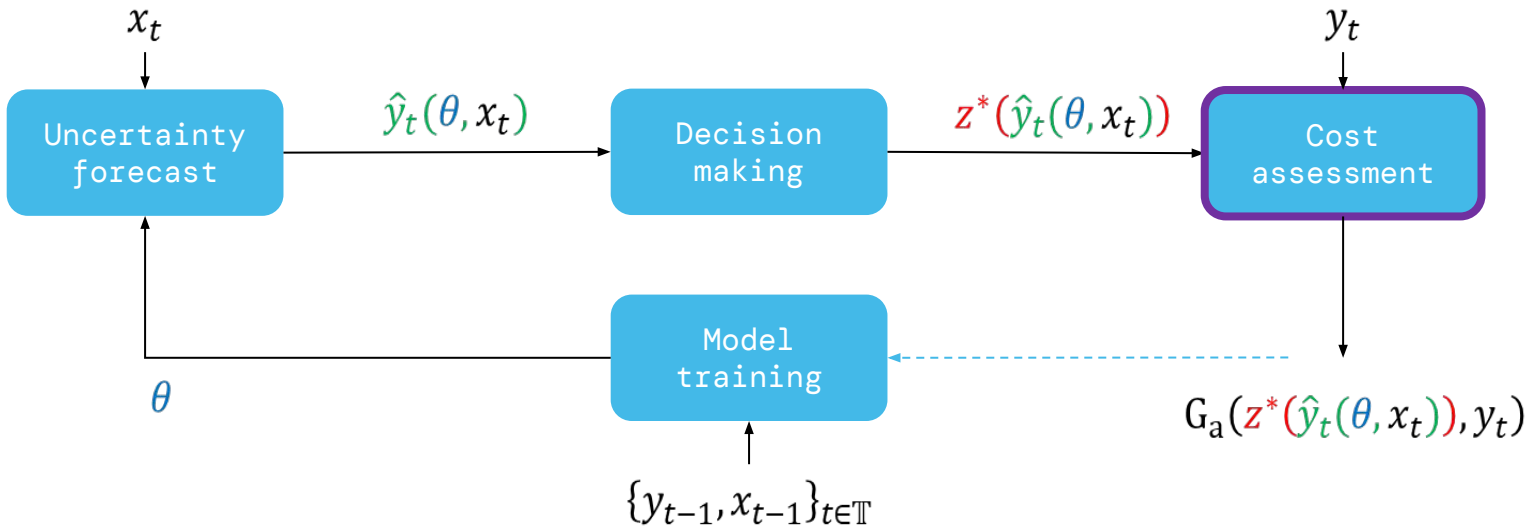
- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

Closing the loop



- Predictive model training
- Load and reserve requirements forecast
- Operation Planning
- Re-dispatch of resources in real time

$$\theta_T \in \arg\min_{\theta \in \Theta, \hat{y}_t, z_t^*} \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \arg\min_{z \in Z} G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

How to solve: Exact Bilevel Formulation

- Bilevel formulation + MPEC reformulation based on KKT conditions

$$\theta_T \in \arg\min_{\theta\in\Theta,\hat{y}_t,z_t^*} \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$
$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \;\forall t \in \mathbb{T}$$
$$z_t^* \in \arg\min_{z\in Z} G_p(z, \hat{y}_t) \;\forall t \in \mathbb{T}$$

+ assuming linear formulation...

$$G_i(z,y) = c_i^\top z + Q_i(z,y)$$
$$Q_i(z,y) = \min_u \{q_i^\top u \mid W_i u \geq b_i - H_i z + F_i y\}$$

$$\min_{\theta\in\Theta,\hat{y}_t,z_t^*,u_t,\pi_t} \frac{1}{T}\sum_{t\in\mathbb{T}} [c_a^\top z_t^* + Q_a(z_t^*, y_t)]$$
$$s.t. \quad \forall t \in \mathbb{T}:$$
$$\hat{y}_t = \Psi(\theta, x_t)$$
$$W_p y_t + H_p z_t^* \geq b_p + F_p \hat{y}_t$$
$$A z_t^* \geq h$$
$$W_p{}^\top \pi_t = q_p$$
$$H_p^\top \pi_t + A^\top \mu_t = c_p$$
$$\pi_t, \mu_t \geq 0$$
$$\pi_t \perp W_p u_t + H_p z_t^* - b_p - F_p \hat{y}_t$$
$$\mu_t \perp A z_t^* - h$$

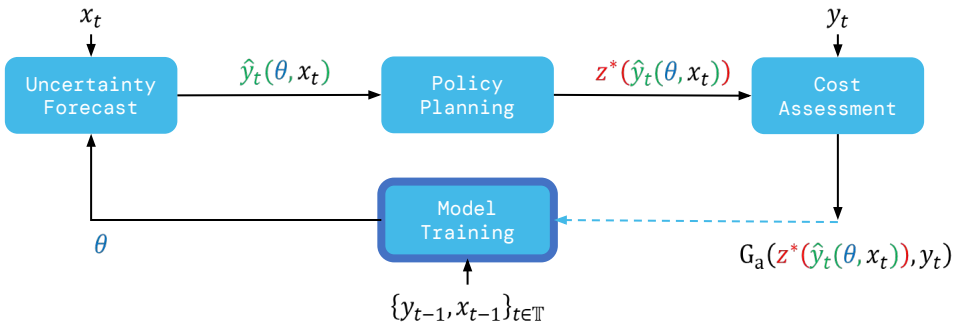
How to solve: Scalable Heuristic

- Generic formulation: Many heuristic and meta-heuristic methods can be applied

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min} G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    Plan Policy: $z_t^* \leftarrow \arg\min_{z \in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t \in \mathbb{T}}(cost_t)$
end

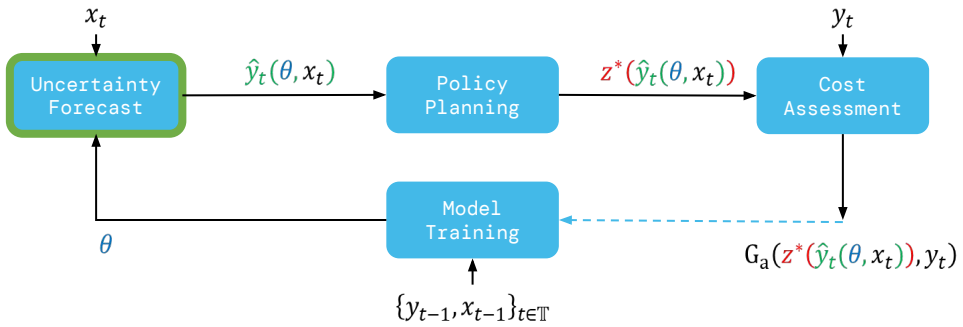
How to solve: Scalable Heuristic

- Generic formulation: Many heuristic and meta-heuristic methods can be applied

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \; \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min} \, G_p(z, \hat{y}_t) \; \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    Plan Policy: $z_t^* \leftarrow \arg\min_{z \in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t \in \mathbb{T}}(cost_t)$
end

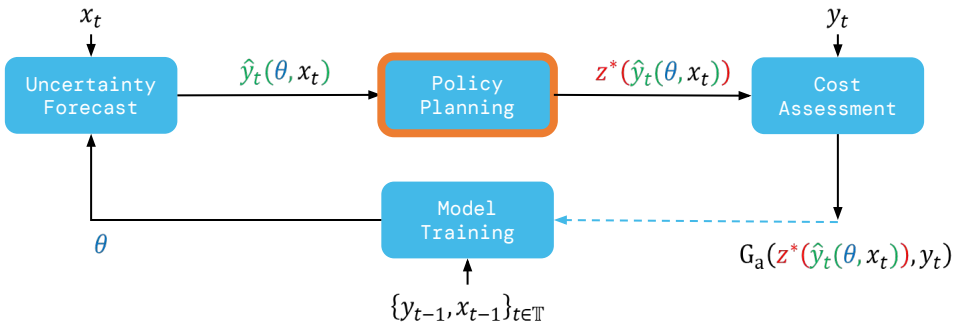
How to solve: Scalable Heuristic

- Generic formulation: Many heuristic and meta-heuristic methods can be applied

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    Plan Policy: $z_t^* \leftarrow \arg\min_{z \in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t \in \mathbb{T}}(cost_t)$
end

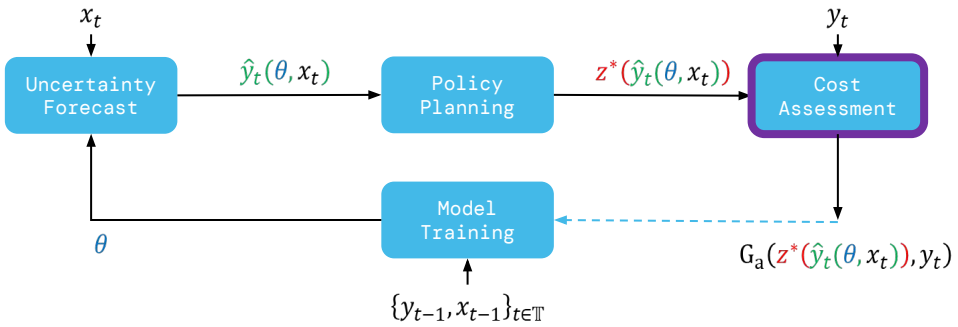
How to solve: Scalable Heuristic

- Generic formulation: Many heuristic and meta-heuristic methods can be applied

$$\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \boxed{\frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)}$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    Plan Policy: $z_t^* \leftarrow \arg\min_{z \in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t \in \mathbb{T}}(cost_t)$
end

How to solve: Scalable Heuristic

Use zero-order methods such as Nelder-Mead to update parameters θ

Pros:

- Simplicity and generalization: As long as we can compute costs efficiently enough, predictive and decision models can assume any form.

Cons:

- Requires repeated cost assessments per each step.
- Curse of dimensionality.

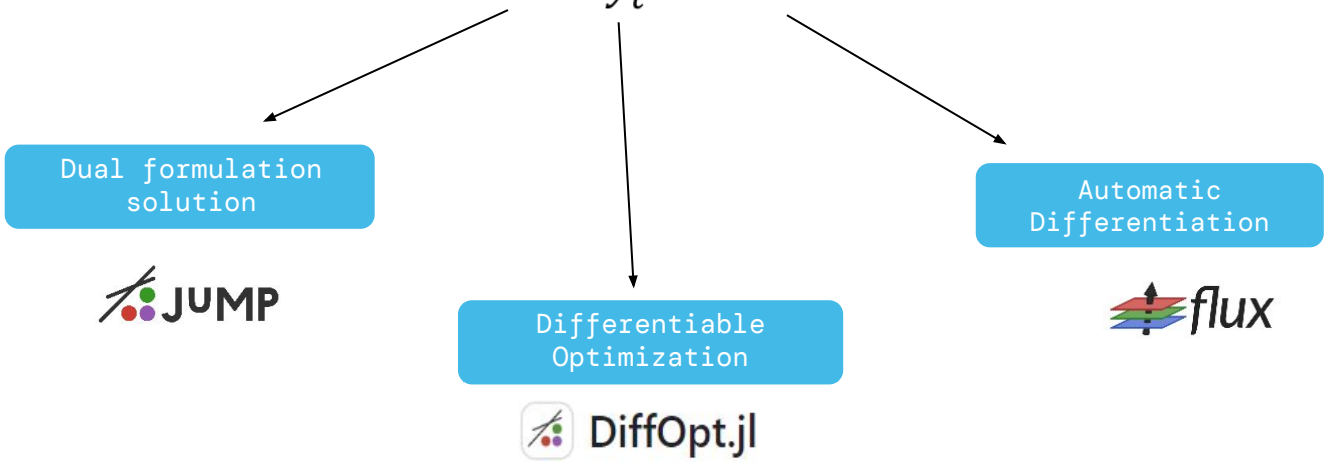
How to solve: Scalable Heuristic + Gradients

Update parameters θ using first-order methods such as gradient descent.

$$C_t = Cost\left(z^*\left(\Psi(\theta, x_t)\right), y_t\right)$$

$$\hat{y}_t = \Psi(\theta, x_t)$$

$$\frac{\partial C_t}{\partial \theta} = \frac{\partial C_t}{\partial z^*} \cdot \frac{\partial z^*}{\partial \hat{y}_t} \cdot \frac{\partial \Psi(\theta, x_t)}{\partial \theta}$$



Study case

- PGLib-OPF: Energy networks library used for benchmarking (Babaeinejadsarookolaee, 2019)
- https://github.com/power-grid-lib/pglib-opf
- Synthetic timeseries
- Auto-regressive forecast models
- https://github.com/LAMPSPUC/ApplicationDrivenLearning.jl/tree/examples/examples/matpower

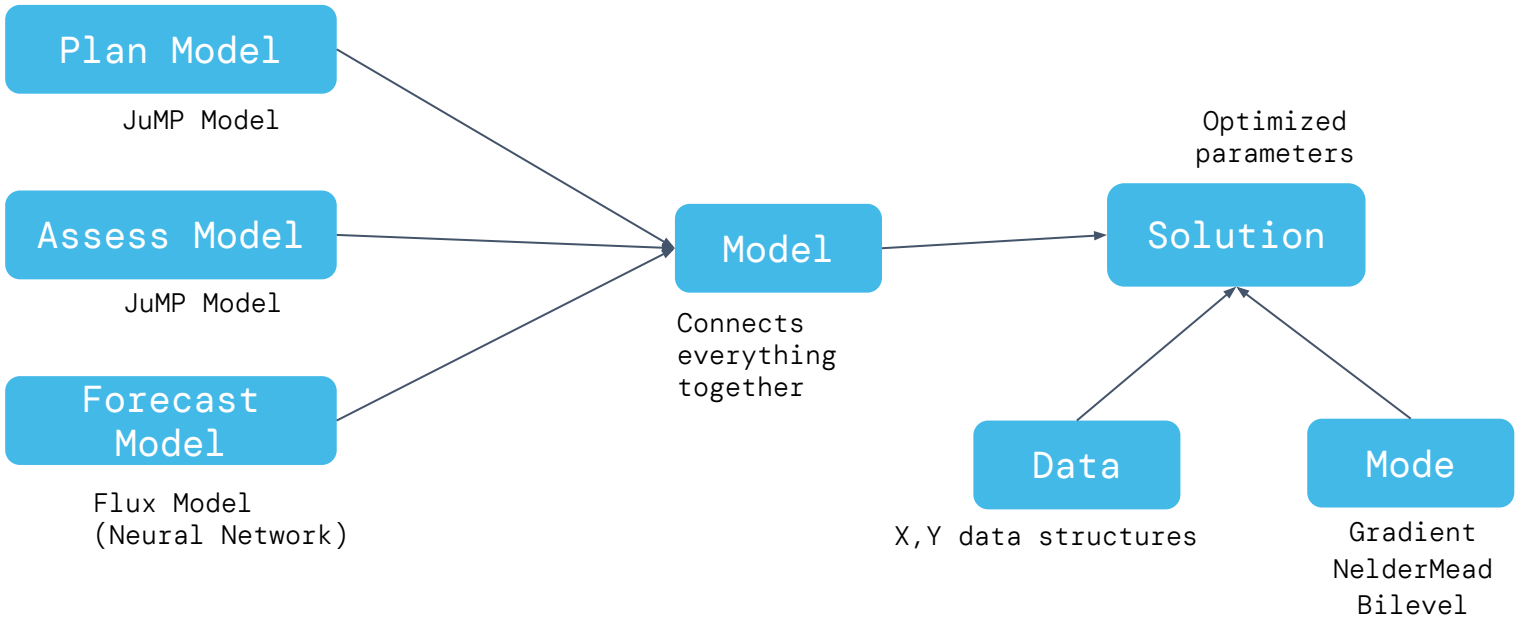
Study case

Out-of-sample costs and relative gain by method and system size (in #buses):

N Buses	LS	MPI-NM	MPI-GD	GD
24	7.608,62	7.617,16 (-0,11%)	7.627,59 (-0,25%)	**7.604,44 (0,05%)**
118	11.515,94	11.177,89 (2,94%)	**10.572,90 (8,19%)**	10.579,62 (8,13%)
179	123.313,02	113.038,81 (8,33%)	**82.549,92 (33,06%)**	90.703,52 (26,44%)
240	801.135,26	-	**760.829,71 (5,03%)**	767.958,60 (4,14%)
300	82.968,84	-	76.625,42 (7,65%)	**76.313,26 (8,02%)**
500	29.154,24	-	**29.051,50 (0,35%)**	29.061,10 (0,32%)
588	33.496,56	-	**29.362,07 (12,34%)**	30.577,54 (8,71%)
793	48.403,34	-	**37.940,31 (21,62%)**	42.558,32 (12,08%)
1354	176.562,56	-	**172.565,10 (2,26%)**	-

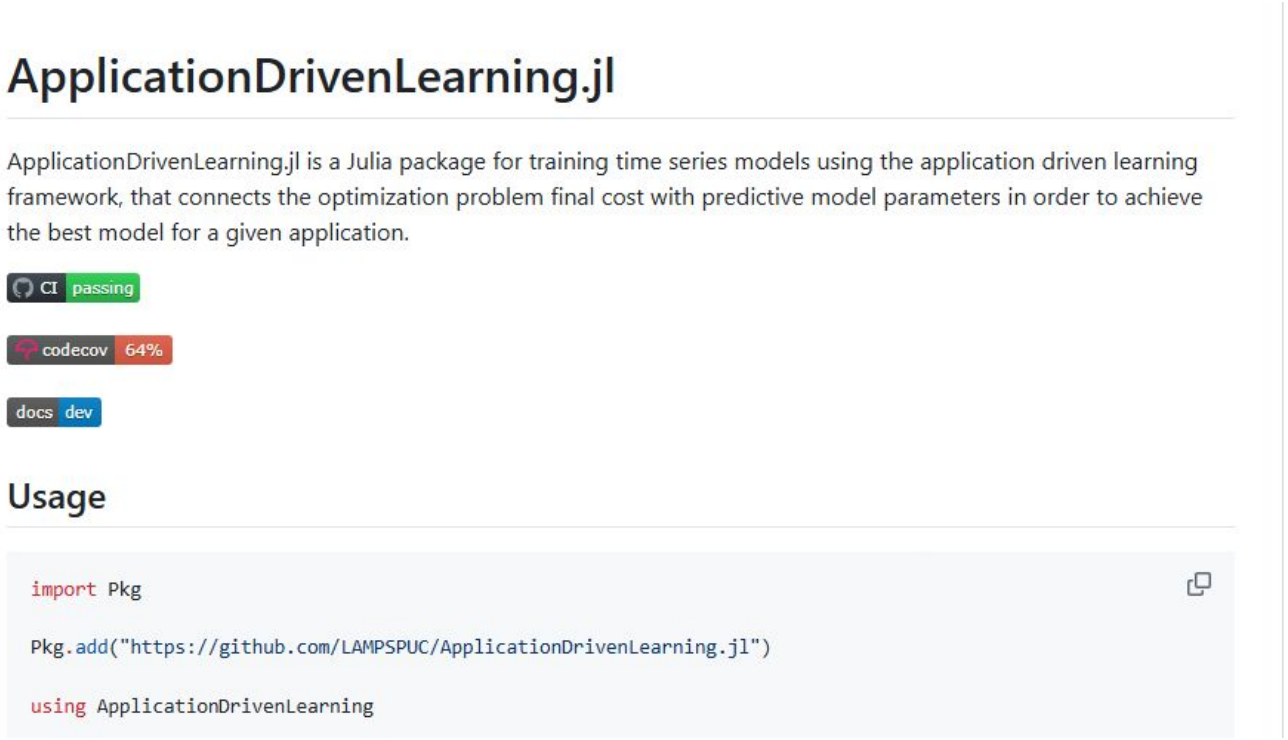
ApplicationDrivenLearning.jl

ApplicationDrivenLearning.jl provides an easy a straightforward way of training models following the closed-loop framework.



ApplicationDrivenLearning.jl

ApplicationDrivenLearning.jl provides an easy a straightforward way of training models following the closed-loop framework.



ApplicationDrivenLearning.jl

ApplicationDrivenLearning.jl is a Julia package for training time series models using the application driven learning framework, that connects the optimization problem final cost with predictive model parameters in order to achieve the best model for a given application.

CI passing

codecov 64%

docs dev

Usage

```
import Pkg

Pkg.add("https://github.com/LAMPSPUC/ApplicationDrivenLearning.jl")

using ApplicationDrivenLearning
```

Small example

```
using JuMP
using Flux
import HiGHS
using ApplicationDrivenLearning

# main model and policy / forecast variables
model = ApplicationDrivenLearning.Model()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)

# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)

# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c3 ≥ 0
    c4 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c3 ≥ 100 * (θ.assess-z.assess)
    c4 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c3 + c4)
```

Small example

```
using JuMP
using Flux
import HiGHS
using ApplicationDrivenLearning

# main model and policy / forecast variables
model = ApplicationDrivenLearning.Model()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)
```

```
# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)
```

```
# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c3 ≥ 0
    c4 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c3 ≥ 100 * (θ.assess-z.assess)
    c4 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c3 + c4)
```

Small example

```
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# data
X = reshape([1 1], (2, 1)) .|> Float32
Y = Dict(θ => [10, 20] .|> Float32)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training the full model
solution = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)

# getting predictions
pred = model.forecast(X')

# extracting solution
println(solution.params)
```

Small example

```
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# data
X = reshape([1 1], (2, 1)) .|> Float32
Y = Dict(θ => [10, 20] .|> Float32)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training the full model
solution = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)

# getting predictions
pred = model.forecast(X')

# extracting solution
println(solution.params)
```

Small example

```
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# data
X = reshape([1 1], (2, 1)) .|> Float32
Y = Dict(θ => [10, 20] .|> Float32)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training the full model
solution = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)

# getting predictions
pred = model.forecast(X')

# extracting solution
println(solution.params)
```

Conclusions

The package:

- https://github.com/LAMPSPUC/ApplicationDrivenLearning.jl
- Friendly interface, specially for those already introduced to JuMP.jl.
- Multiple solution modes already implemented.
- High performance solution methods already implemented.

- The method
  - Demonstrates performance gains compared to the open-loop methodology.
  - Incorporates application's incentives directly to forecasts.
  - Potential gains even in large-scale systems.
  - Ready to be used in practice (by academia and industry).

Next steps

- Generalize assessment cost function to arbitrary variable expressions (maybe simulations outputs).
- Expand available solution methods (genetic algorithms, ensembles,...)
- Methods and tweaks to improve convergence and results (e.g. ridge regularization).
- Stochastic version (probabilistic forecasts).

Thank you very much!

Github: Giovanni3A

email: giovanni@soma.energy