Recurrent Machines for Likelihood-free Inference

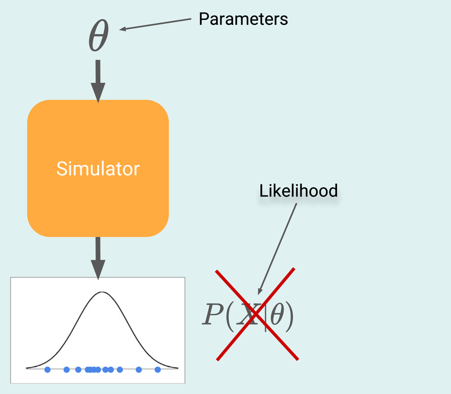
Arthur Pesah • Antoine Wehenkel • Gilles Louppe

University of Liège, Belgium

Likelihood-free inference

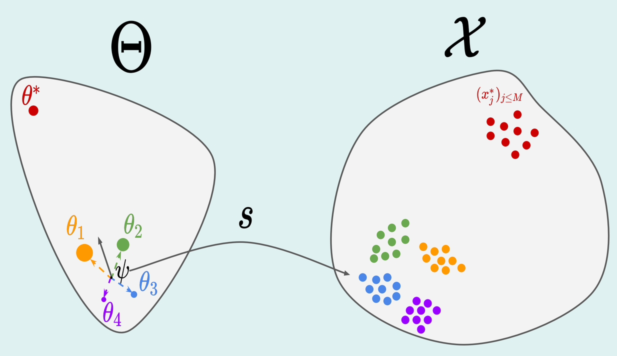
Principle

Goal: to find the parameters of a stochastic simulator (or generator) best fitting some real data.
Constraint: the probability distribution of the results conditionnaly to the parameters (likelihood) is unknown.



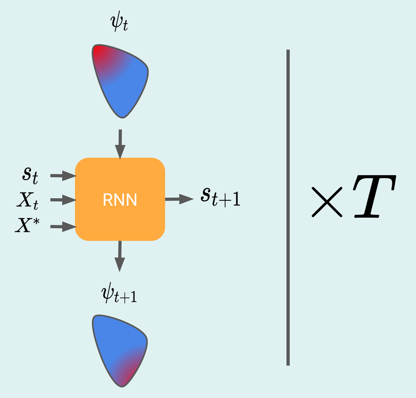
General algorithm

1. Generating several parameters using a **proposal distribution**.
2. Performing the simulation with those parameters.
3. Moving the proposal distribution by **comparing the generated data with the real ones.**



Contribution

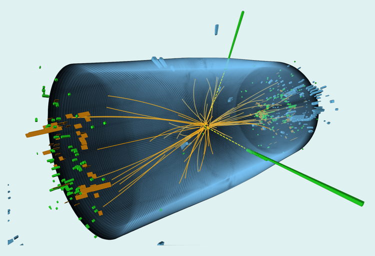
Learning the step **3.** of the general algorithm, improving the meta-learning framework by showing that it can be used to learn an optimization procedure when no explicit objective value is available at each step.



Applications

Physics

Simulators: Detectors output after particle collisions ; evolution of the universe under a cosmological model.
Parameters: constants of a physical theory (e.g.: coupling constants in a Lagrangian).

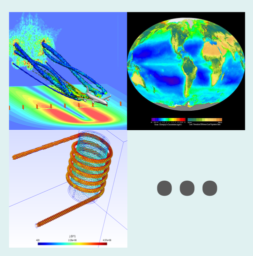


Life Sciences

Simulator: evolution of populations ; dynamics of the potential inside neurons.
Parameters: constants of the differential equations modelling the evolution.

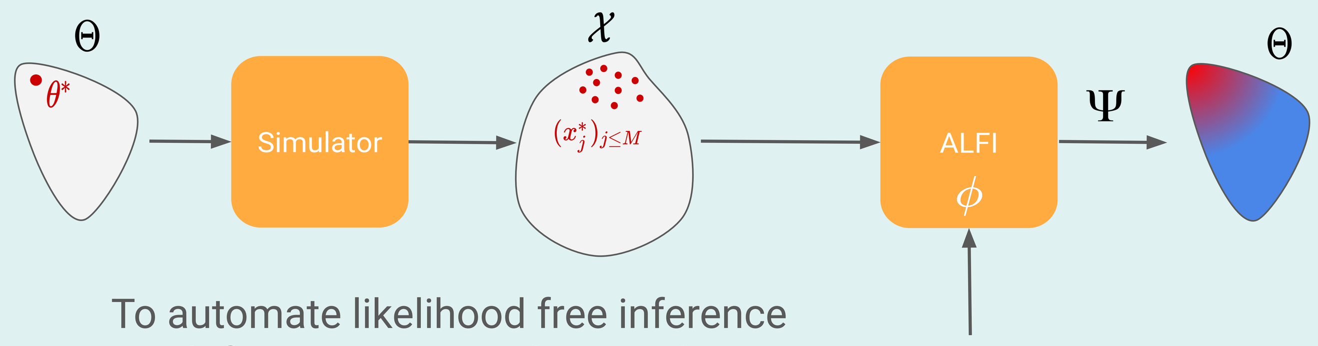
Everything Simulable

Simulator: A procedure to generate data conditionnaly to some parameters.
Parameters: The input of the simulator. LFI aims at finding parameters that will match the data generated and the data observed as best.



Architecture

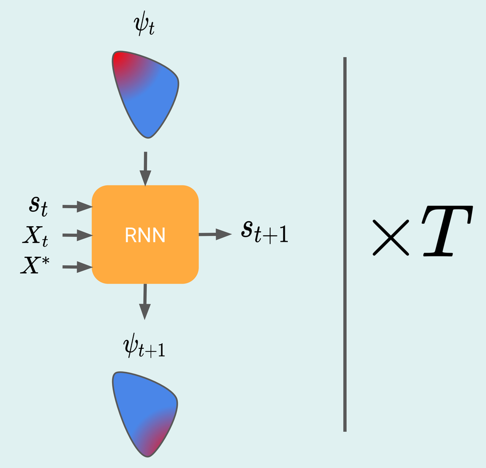
General Architecture



To automate likelihood free inference we define ALFI, a neural network aiming at inverting the simulation process.

Parametrized (ϕ) and differentiable

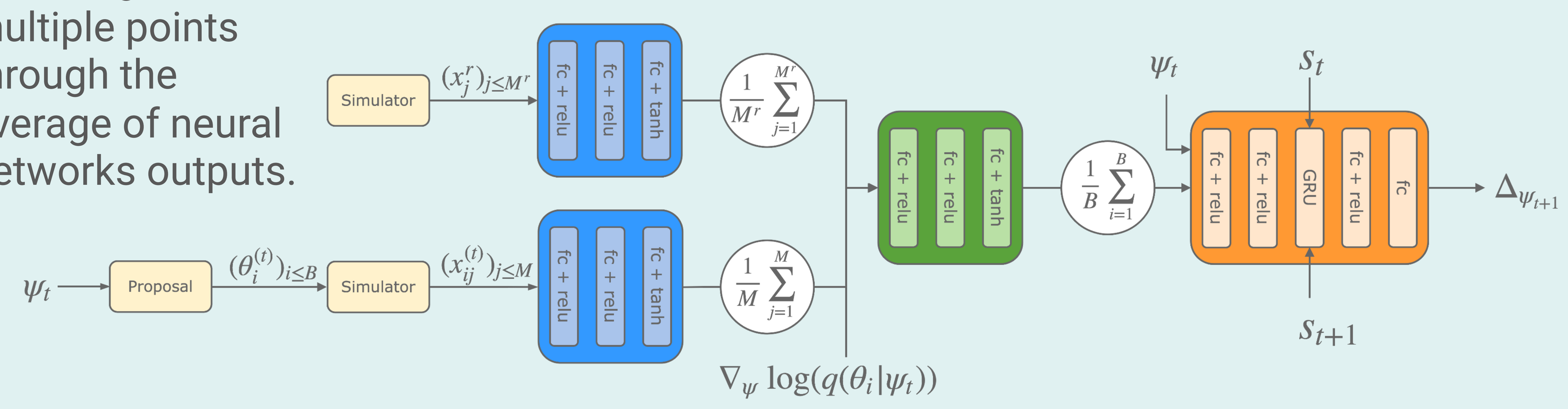
RNN-based recurrent machine



A recurrent neural network is used to invert the simulator incrementally.

ALFI: Automatic Likelihood-Free Inference

Encoding of multiple points through the average of neural networks outputs.



Meta-learning comes into play

Learning to learn

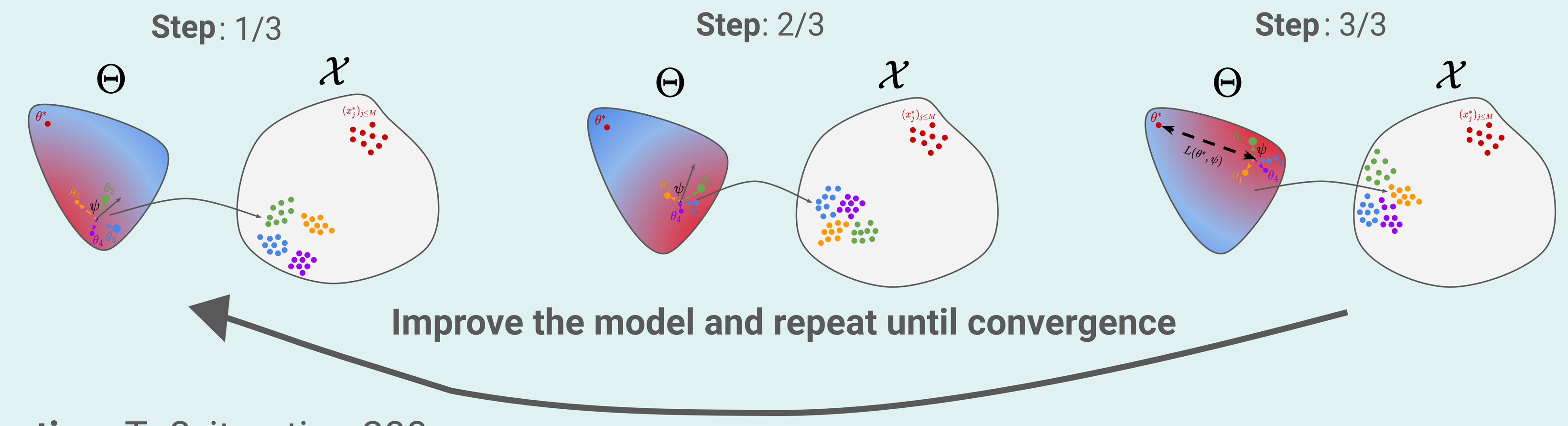
Previous work: an RNN is trained to deduce at each step **the best descent direction**, from either the value of the objective function or its gradient. During the meta-training, **the values of the objective function at each step is used as an error signal** to update the weights of the RNN appropriately.

Our framework: the RNN is trained to deduce at each step the **best update for the proposal distribution**, from the sampled parameters, the data generated by the simulator, and the real data. The RNN learns by itself to compare the generated data with the real ones and to move the proposal towards the most accurate parameters. During the meta-training, **the true parameter is known** and can be used to update the weights of the RNN appropriately.

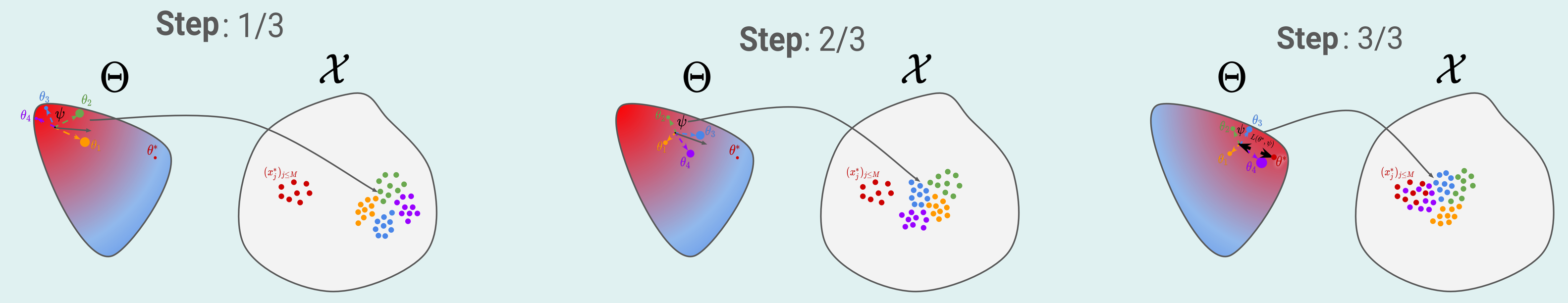
Meta-algorithm

Algorithm
1. Create many artificial "true parameters" (and corresponding simulated data).
2. For each true parameter, run the RNN with T steps and get the output proposal distribution.
3. Evaluate a loss representing a distance between the proposal and the true parameter.
4. Backpropagate.

Training example: T=3, iteration 0



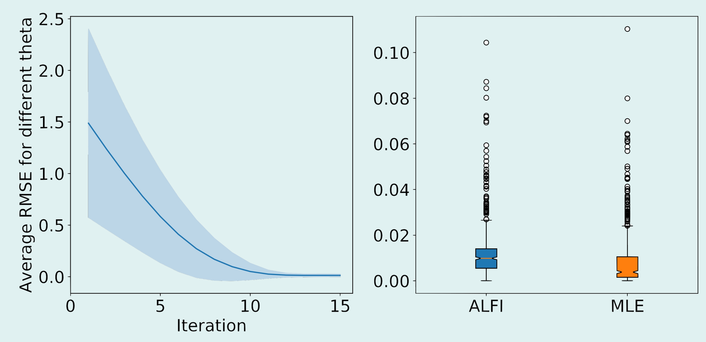
Testing: T=3, iteration 200



Results

Poisson

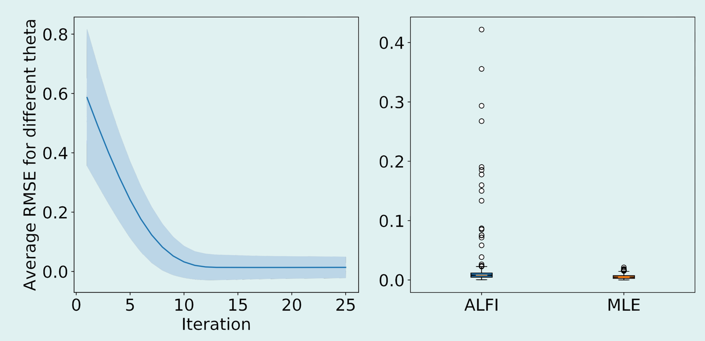
Simulator: Poisson distribution.



Left: root mean-squared error (RMSE) between the true parameter and the predicted one (expectancy of the proposal distribution), at each step of the RNN (T=15), averaged over several different true parameters.
Right: distribution of the error over different true parameters. Comparison with the maximum likelihood estimator (MLE).

Linear Regression

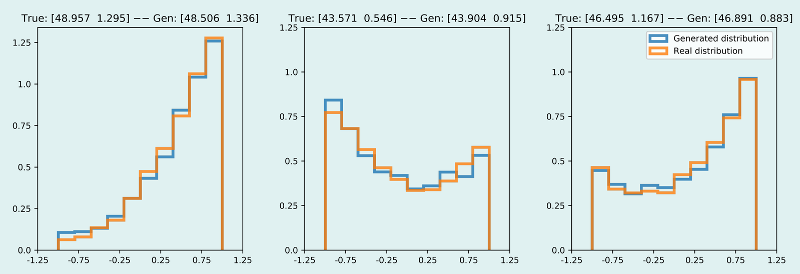
Simulator: Linear model with Gaussian noise.



Left: root mean-squared error (RMSE) between the true parameter and the predicted one (expectancy of the proposal distribution), at each step of the RNN (T=15), averaged over several different true parameters.
Right: distribution of the error over different true parameters. Comparison with the maximum likelihood estimator (MLE).

Simplified Physics Simulator

Simulator: simplified model of electron-positron collisions resulting muon-antimuon pairs. Takes two parameters (Fermi constant and beam energy) and gives the scattering angle of the muons.



Comparison between the distribution of the real observations, and the one produced by the observations generated by the predicted parameter

Future Works

Can it scale to higher dimensional problem?

What is learned by ALFI ?

Does this architecture have transfer learning capabilities?

Is the procedure learned similar to handcrafted LFI methods ?

artix41
arthur.pesah@gmail.com

WehenkelAntoine
antoine.wehenkel@uliege.be

glouppe
g.louppe@uliege.be