First building block : ABC simulations

- Compute summary statistic \( \mu \) for each simulation
- Given a certain model, perform \( n \) simulations, each with a parameter drawn from the prior distribution
- Based on a distance \( d(\hat{\mu}, \mu) \) and a tolerance \( \varepsilon \), decide whether the summary statistic value is close enough to the corresponding value on observed data
- We store all selected simulations (parameters and summary statistics) in a reference table.

Second building block : Random Forests

CART

Random Forests are based on a CART, Classification and Regression Trees, algorithm [4].

A CART is a machine learning algorithm whose principle is to partition the predictor space into disjoint subspaces, in an iterative manner, and each one is assigned a prediction value which will be used for test data falling in this subspace. Once the partitioning is done, we have a binary tree structure which could predict outcomes from an input data, either classes or continuous values.

First building block : ABC simulations

Given an observed data, the basic idea of ABC, Approximate Bayesian Computations [1], is to approximate the likelihood of a parameterized model with selected simulations, by comparing the observed data and simulated ones using computed summary statistics. The table of summary statistics for simulated data is called the reference table.

ABC posterior methodologies

Model choice: Simulate data for several models and choose the best model to fit our data
Parameter estimation: Simulate data for one model and infer one or several parameters for this model given the observed data

A sensible workflow is to first choose a model and then infer its parameters.

- Compute simulations with several models, and the reference table with model-indexed lines using a simulator (DIYABC, PyABC etc.)
- Apply Model Choice Methodology with AbcRanger
- Apply Parameter Estimation Methodology with AbcRanger

Challenges of ABC

Computational challenges with ABC/Random Forests

With 100 000 lines and more than 10 000 summary statistics, each tree could reach over 3 gigabytes of memory size. Typically we need 500 or 1000 trees for good prediction performance, so, even with state of the art RF packages like [6], memory constraints are preventing completion of the training.

A new implementation of Random Forest for ABC

Since ABC procedures only use trained Random Forests on a known set of observations, we have altered the random forest training computation by using only a subset of in-memory trees at a time and accumulating the required outcomes (predictions and statistics). Memory footprint is vastly improved and there is no performance cost.

Our solution

Random Forests [5] are a three pronged extension of CART:

- Ensemble method: Training a set of CART (not just one), and getting the majority vote (resp. mean) for classification (resp. regression)
- Bootstrapping: Training data is random sampled (with replacement) for each tree
- Feature bagging: At each node of a growing tree, find the best split on a random subset of the features

Advantages in an ABC setting:

- robust to noise
- almost free variable importance
- free (out-of-bag) cross-validation procedure
- easy parallelization
- good scaling properties (samples and features)
- classifier and regressor (both are used)

Computational challenges with ABC/Random Forests

Random Forests [5] are a three pronged extension of CART:

- Ensemble method: Training a set of CART (not just one), and getting the majority vote (resp. mean) for classification (resp. regression)
- Bootstrapping: Training data is random sampled (with replacement) for each tree
- Feature bagging: At each node of a growing tree, find the best split on a random subset of the features

Advantages in an ABC setting:

- robust to noise
- almost free variable importance
- free (out-of-bag) cross-validation procedure
- easy parallelization
- good scaling properties (samples and features)
- classifier and regressor (both are used)

References

[9] A new implementation of Random Forest for ABC.