Figaro 3.0 Release Notes

About Figaro

Reasoning under uncertainty requires taking what you know and inferring what you don't know, when what you know doesn't tell you for sure what you don't know. A well-established approach for reasoning under uncertainty is probabilistic reasoning. Typically, you create a probabilistic model over all the variables you're interested in, observe the values of some variables, and query others. There is a huge variety of probabilistic models, and new ones are being developed constantly. Figaro is designed to help build and reason with the wide range of probabilistic models.

Developing a new probabilistic model normally requires developing a representation for the model and a reasoning algorithm that can draw useful conclusions from evidence, and in many cases also an algorithm to learn aspects of the model from data. These can be challenging tasks, making probabilistic reasoning require significant effort and expertise. Furthermore, most probabilistic reasoning tools are standalone and difficult to integrate into larger programs.

Figaro is a probabilistic programming language that helps address both these issues. Figaro makes it possible to express probabilistic models using the power of programming languages, giving the modeler the expressive tools to create all sorts of models. Figaro comes with a number of built-in reasoning algorithms that can be applied automatically to new models. In addition, Figaro models are data structures in the Scala programming language, which is interoperable with Java, and can be constructed, manipulated, and used directly within any Scala or Java program.

Figaro is extremely expressive. It can represent a wide variety of models, including:

- directed and undirected models
- models in which conditions and constraints are expressed by arbitrary Scala functions
- models involving inter-related objects
- open universe models in which we don't know what or how many objects exist
- models involving discrete and continuous elements
- models in which the elements are rich data structures such as trees
- models with structured decisions
- models with unknown parameters

Figaro provides a rich library of constructs to build these models, and provides ways to extend this library to create your own model elements.

Figaro's library of reasoning algorithms is also extensible. Current built-in algorithms include:

- Exact inference using variable elimination
- Belief propagation
- Lazy factored inference for infinite models

- Importance sampling
- Metropolis-Hastings, with an expressive language to define proposal distributions
- Support computation
- Most probable explanation (MPE) using variable elimination or simulated annealing
- Probability of evidence using importance sampling
- Particle Filtering
- Factored frontier
- Parameter learning using expectation maximization

Figaro provides both regular (the algorithm is run once) and anytime (the algorithm is run until stopped) versions of some of these algorithms. In addition to the built-in algorithms, Figaro provides a number of tools for creating your own reasoning algorithms.

Figaro is free and is released under an open-source license (see license file). The public code repository for Figaro can also be found at https://github.com/p2t2

What's New in Figaro 3.0?

Many new features have been introduced into Figaro 3.0 since Figaro 2.0 was released. These include:

- A collections library. This library includes support for processes defined over an infinite space, such as Gaussian processes or continuous-time Markov processes. It also supports collections consisting of an unknown number of elements. Many convenient methods are defined on Figaro collections, including mapping values of elements in a collection through a function and folds or aggregates (for finite collections).
- A number of new algorithms:
    - Lazy variable elimination allows variable elimination to be used on very large or infinite models by only expanding part of the model and quantifying the contribution of the unexpanded part of the model on the query.
    - Belief propagation for approximate factored inference.
    - The factored frontier algorithm for filtering.
    - A wider variety of expectation-maximization-based learning algorithms.
- Improvements to existing algorithms:
    - Forward sampling algorithms like importance sampling and particle filtering now work in log space to avoid underflow.
    - Importance sampling now properly implements likelihood weighting to avoid too many rejections.
    - More algorithms support computing probability of evidence, including variable elimination, importance sampling, and particle filtering.
    - Observations on continuous elements are now implemented as constraints on the arguments of the elements, which helps avoiding rejections and mixing.
    - Ability to sample particular target variables without sampling all the variables.
- Features to make programming easier:

    - New patterns for learning. Parameter collections enable you to reuse the same parameters in both the training model and operational model.
    - One line shortcuts for common queries.
    - Ability to query the mean and variance of a Double element.
    - Ability to add logarithmic constraints to elements.
- New element classes in the library:
    - Multivariate normal
    - Inverse gamma
    - Parameterized binomial
    - Fold for implementing associative aggregates efficiently
- Numerous minor improvements and bug fixes

Version Notes

New in Version 3.1.0 (??-???-2015)

Requirements

- Scala 2.11 is required to run Figaro
    - Tested against Scala 2.11.4
- Simple Build Tool (SBT) 0.13.6

New Features

- No changes

Improvements and Changes

- No changes

Bug Fixes

- No changes

Examples

- No changes

New in Version 3.0.0 (09-JAN-2015)

Requirements

- Scala 2.11 is required to run Figaro
    - Tested against Scala 2.11.4
- Simple Build Tool (SBT) 0.13.6

New Features

- Make factors for FoldLeft (#330)

Improvements and Changes

- Better Figaro collections (#258)
- Infinite HMM example for LFI (#272)
- Unit tests take too long (#278)
- observe() method for continuous distributions (#287)

Bug Fixes

- No changes

Examples

- No changes

New in Version 2.5.0 (11-NOV-2014)

Requirements

- Scala 2.11 is required to run Figaro
    - Tested against Scala 2.11.2
- Simple Build Tool (SBT) 0.13.6

New Features

- Initial Particle BP implementation (#308)
- Provide parameter collection pattern for easier model definition (#316)

Improvements and Changes

- More efficient implementation of getting statistics in GeneralizedEM (#220)
- Allow learning algorithms to learn all parameters without having them specified explicitly (#227)
- EM should include an early termination criteria (#253)
- Create an AssertEvidence object whose apply asserts evidence on the default universe (#264)

Bug Fixes

- Fixes to MHTest (#310)

Examples

- Convert FairDice and FairCoin examples to use new learning abstraction (#311)

New in Version 2.4.0 (16-OCT-2014)

Requirements

- Scala 2.11 is required to run Figaro
    - Tested against Scala 2.11.2
- Simple Build Tool (SBT) 0.13.5

New Features

- No changes

Improvements and Changes

- Layers is too expensive (#209)
- Allow parameterized elements to use multiple parameters (#256)
- Minor changes to BP code (#294)
- Getting density of Atomic elements outside of distribution support (#297)
- Many changes to support Particle BP (#298)

Bug Fixes

- MH Performance Problems (#150)
- Does BP create factors too soon? (#224)
- CompoundTest is failing with a NoSuchElementException (#255)
- MultiValuedTest is failing (#279)
- Unweighted sampler doesn't correctly record the first and last samples (#288)

Examples

- No changes

New in Version 2.3.0 (16-SEP-2014)

Requirements

- Scala 2.11 is required to run Figaro
    - Tested against Scala 2.11.2
- Simple Build Tool (SBT) 0.13.5

New Features

- Factored Frontier Algorithm for DBNs (#226)
- Infer and learn abstraction (#243)

Improvements and Changes

- Importance sampling should implement probability of evidence (#174)
- Require filtering algorithms to take a transition function from a previous and static universe (#247)
- Variable elimination order may be non-optimal (#276)

Bug Fixes

- Remove System.gc from ParticleFilter (#38)
- Resolve warnings (#67)
- Eliminate deprecation warnings in Scala 2.11 (#208)
- Factors are created incorrectly for models in multiple universes (#231)
- VE bug in old SingleDecision (#233)
- Variable elimination cost computation overflows (#275)
- Lazy BP fails when computing distribution of element with * in its range (#277)

Examples

- No changes

New in Version 2.2.1 (13-JUN-2014)

Requirements

- Scala 2.10 or 2.11 is required to run Figaro
    - Cross-compiled to support Scala 2.10 and 2.11
    - Tested against Scala 2.10.4
    - Tested against Scala 2.11.1
- Simple Build Tool (SBT) 0.13.5

New Features

- No changes

Improvements and Changes

- Likelihood weighting improvements (#203)
- Importance sampling should clear temporaries (#202)
- Sufficient statistics factors should allow anything for which a ProbFactor can be made (#199)
- ParameterizedBinomial (#196)
- Add multi-variate normal distribution (#118)
- EM with more algorithms (#60)

Bug Fixes

- MH does not work when interval is more than 1 (#171)

Examples

- No changes

New in Version 2.2 (06-JUN-2014)

Requirements

- Scala 2.10 or 2.11 is required to run Figaro
    - Cross-compiled to support Scala 2.10 and 2.11
    - Tested against Scala 2.10.4
    - Tested against Scala 2.11.1
- Simple Build Tool (SBT) 0.13.5

New Features

- Figaro tutorial now in LaTeX format (#102)

Improvements and Changes

- Query joint probabilities in factored algorithms (#88)
- Non-loopy chain factors (#92)
- Integrate belief propagation with lazy factored inference (#106)
- Binomial density method is very slow (#131)
- Poisson density method is slow (#132)
- Particle filter should compute probability of evidence (#137)
- Belief propagation should use divide when sending messages to neighbors (#142)
- Using ElementCollection.get in particle filtering grows the size of static universes (#154, partial)
- Exposed the getElementByReference function in ElementCollection (#165)
- Logarithmic constraints (#166)
- Likelihood weighting (#172)
- Updated Annealing to store the old constraints like regular Metropolis Hastings (#177, #178)
- Shortcuts for common queries (#179)
- Cross-compile to support Scala 2.10 and Scala 2.11 (#182)
- Turn distribution into an element (#185)

Bug Fixes

- Fix forward sampling so it samples all active elements instead of all permanent elements (#117)
- Disallow non-caching chains from using factored algorithms (#125)
- Apply factors with * (#126)
- Caching chains cache size (#127)
- Belief propagation underflow protection (#128)
- Parent values changing during evaluation of chain values (#129)
- Clearing lazy values needs to wipe all cached values (#130)
- toNameString can cause crashes (#133)
- Lazy values and abstractions (#134)

- Metropolis Hastings and non-caching chains (#135)
- Delete ExpandTest (#143)
- Move probability of evidence sampler to log space (#147)
- Added evidence computation to belief propagation (#152)
- Remove binary files from GitHub (#158)
- Passing lists of targets to Importance and Metropolis Hastings does not work properly (#181)
- Particle filter in log space (#175)
- Check correctness of BetaParameter and DirchletParameter (#192)
- Do we really need AtomicBetaParameter and AtomicDirichletParameter? (#193)

Examples

- Update examples to reflect #193 changes (#195)

New in Version 2.1 (17-FEB-2014)

Requirements

- Scala 2.10.x is required to run Figaro
    - Tested against Scala 2.10.1
- Simple Build Tool (SBT) 0.13.1
    - Preferred tool for compiling and running unit tests
    - Ant build files have been removed

New Features

- Enabled factor algorithms for IntSelector; added makeValues and makeFactors. (#74)
- Belief propagation. (#61, #109)
- Lazy factored inference. (#65, #91, #104)

Improvements and Changes

- Elements no longer self-generate on initialization. (#47)
- Added hasRef() function to element collections that allows one to check if a reference is resolvable on that collection. (#48)
- Removed inefficient dependencies from Algorithm subclasses. (#51)
- Simple Build Tool (SBT) is now preferred Figaro build tool. (#57, #75, #79)
- Element is now a monad. (#59)
- Removed deprecation warnings. (#67)
- Registered Maps in Universe now take Sets. (#83)

Bug Fixes

- Element collections now check through all possible resolutions of a reference before deciding that it cannot apply any evidence to a reference. (#45)
- Importance sampling now samples temporary elements in chains. (#46)
- Semiring test now manages memory better. (#71)
- Contingent Constraints are now applied correctly in Metropolis Hastings (#82)
- Erroneous reference to Expand.scala removed. (#98, #99)

Examples

- No changes

New in Version 2.0 (10-OCT-2013)

Requirements

- Scala 2.10.x is now required to run Figaro.

New Features

- Figaro now has the ability to represent decisions and contains several algorithms for computing optimal decisions in a model.
- Learning the parameters of elements based on observed evidence has been added to Figaro.
- Expectation maximization algorithm added to Figaro to support parameter learning
- Simulated annealing has been added to compute the most likely state of a model.
- Added basic reflection capability for Figaro elements

Improvements and Changes

- Probability of evidence can now make a distinction between conditions and constraints on elements and additional named evidence applied
- Particle Filtering
    - Now takes an initial universe and a transition function from a universe to universe instead of a stream of universes
    - Improved memory management enables old universes to be cleaned up after the process has moved on to later time steps
- Universe changes
    - Universe now collects conditioned and constrained elements so algorithms do not need to check all elements for satisfaction
    - Calling used and usedBy on elements now computes the element dependencies as needed, but caches the result until the model changes, resulting in more reliable operation and less space usage
    - Removed context control from Universe and moved to the elements themselves
    - Added registration of algorithms on universes so that algorithms can be killed when universe is cleaned up
    - Added the ability to generate a value for all elements in a universe in order
- Chain changes
    - There are no longer two implementations of Chain. Every chain is instantiated with a fixed cache size. Caching chains use a cache of 1000 and non-caching use a cache of 1
    - The type of chain (caching/non caching) is now automatically determined when instantiating from the object Chain
    - Chains now control clearing elements created in their context
    - Two parent chains now changed from nested chains to creating a tuple of the parents into a single parent chain
- Names and references

    - getElement in ElementCollection now changed to getElementByReference, to make clear that the reference is resolved to get a particular reference
    - Reference elements and aggregate elements now work with Variable Elimination
    - Improvements supporting reasoning about objects in class hierarchies
        - Multiple elements with the same name are now allowed. Getting the element by reference returns the most recent one
        - NamedEvidence is applied to all current and future elements with the name
- MakeList
    - Does not inherit from Chain anymore
    - Now works in Variable Elimination
- Setting the seed of the Figaro random number generator now creates deterministic inference
- Elements whose values have been explicitly set or observed do not have their values regenerated

Bug Fixes

- Fixed a bug that prevented factors from taking into account latest evidence when a new factored algorithm is created
- Fixed bug where importance sampling was sampling from inactive elements
- Fixed Metropolis-Hastings bug where it would not run when there were no stochastic elements in the initial state
- Fixed Metropolis-Hastings disable initialization during test mode
- Fixed Metropolis-Hastings bug that was erroneously recording state in non-satisfied conditions
- Fixed bug in CPD where temporary parent tuple not created on the same universe

Examples

- Examples now kill their algorithms when they are done
- Examples added for decisions, parameter learning and hierarchical reasoning