Introducing the Bayesian analysis toolkit and pypmc

Frederik.Beaujean@lmu.de
Excellence cluster universe

Belle II workshop, Karlsruhe, Feb 2015

BAT men: A. Caldwell, D. Greenwald (Belle I/II), D. Kollár, K. Kröninger

pypmc: S. Jahn

Session Overview

F. Beaujean

Bayes: basics, numerics

C. Bobeth

global fit of rare B decays: motivation, results

D. van Dyk

computing observables with EOS

Questions

1. Given the data from Belle, BaBar, LHCb, (… Belle II), what are the likely values of Wilson coefficients, or CKM phases, or …?
2. If there is a deviation from the SM, which NP model is preferred?

Bayes' theorem

Parameter inference

$$\boxed{P(\theta|D, M) = \frac{P(D|\theta, M)P_0(\theta|M)}{P(D|M)}}$$

posterior $\propto$ likelihood $\times$ prior

Bayes' theorem

Parameter inference

$$\boxed{P(\boldsymbol{\theta}|D, M) = \frac{P(D|\boldsymbol{\theta}, M)P_0(\boldsymbol{\theta}|M)}{P(D|M)}}$$

posterior $\propto$ likelihood $\times$ prior

Model comparison

$$\text{evidence } P(D|M) = \int \mathrm{d}\boldsymbol{\theta}\, P(D|\boldsymbol{\theta}, M)P_0(\boldsymbol{\theta}|M)$$

$$\boxed{\frac{P(M_1|D)}{P(M_2|D)} = \frac{P(D|M_1)}{P(D|M_2)} \times \frac{P(M_1)}{P(M_2)}}$$

posterior odds $=$ Bayes factor $\times$ prior odds

Parameters

- $\boldsymbol{\theta} =$ Wilson coefficients, CKM parameters, theory uncertainties, detector effects …
- $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\nu})$ parameters of interest + nuisance

Applying Bayes' theorem

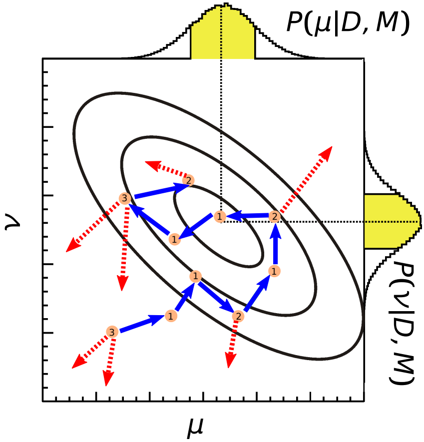
Parameters

- $\theta =$ Wilson coefficients, CKM parameters, theory uncertainties, detector effects ...
- $\theta = (\mu, \nu)$ parameters of interest + nuisance

Integration

- marginalization $P(\mu|D, M) = \int \mathrm{d}\nu \, P(\mu, \nu|D, M)$
- evidence $P(D|M) = \int \mathrm{d}\theta \, P(D|\theta, M) P_0(\theta|M)$
- curse of dimensionality

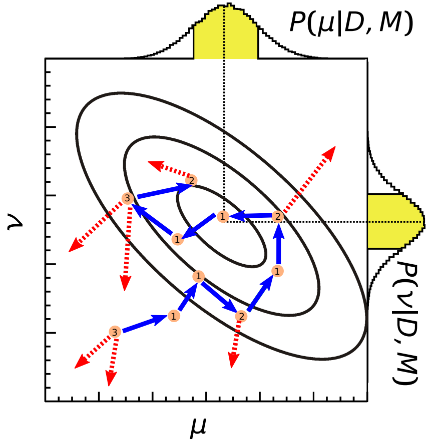
$\Rightarrow$ need samples from posterior



METROPOLIS HASTINGS ALGORITHM

one sample for each step

1. propose move
2. accept or stay



METROPOLIS HASTINGS ALGORITHM

one sample for each step

1. propose move
2. accept or stay

- sample near mode ⇒ seed for optimization
- uncertainty propagation $f(\mu, \nu) \to P(f|D, M)$

BAYESIAN ANALYSIS TOOLKIT

- home page http://mpp.mpg.de/bat
- fork me on https://github.com/bat/bat
- first release in 2008, latest 0.9.4.1 in Jan 2015

Bayesian Analysis Toolkit

- home page http://mpp.mpg.de/bat
- fork me on https://github.com/bat/bat
- first release in 2008, latest 0.9.4.1 in Jan 2015

Motivation

- reinventing the wheel time waster, error prone
- create C++ toolkit to supply algorithms/models, so user can focus on the problem

BAYESIAN ANALYSIS TOOLKIT

- home page http://mpp.mpg.de/bat
- fork me on https://github.com/bat/bat
- first release in 2008, latest 0.9.4.1 in Jan 2015

MOTIVATION

- reinventing the wheel time waster, error prone
- create C++ toolkit to supply algorithms/models, so user can focus on the problem

FEATURES

- implemented: MCMC (multithreaded), simulated annealing ...
- depends on ROOT: I/O, plots, optimization (Minuit) ...
- optional: roostats, CUBA (integration)
- docs, tutorials, examples ... on web page

Components

$$\boxed{P(\theta|D, M) \propto P(D|\boldsymbol{\theta}, M)P_0(\boldsymbol{\theta}|M)}$$

User defined

- create model
- read data

Components

$$\boxed{P(\theta|D,M) \propto P(D|\theta,M)P_0(\theta|M)}$$

User defined

- create model
- read data

Define MyModel : BCModel

- `AddParameter("mu", 0, 1)`
- `LogLikelihood()`
- `LogAPrioriProbability()`

Read data

- text file
- ROOT tree
- anything in C++

$$\boxed{P(\theta|D,M) \propto P(D|\theta,M)P_0(\theta|M)}$$

User defined

- create model
- read data

Common tools

- `Normalize()`
- `FindMode()`
- `MarginalizeAll()`
- `PrintAllMarginalized()`
- `PrintKnowledgeUpdatePlots()`

Define MyModel : BCModel

- `AddParameter("mu", 0, 1)`
- `LogLikelihood()`
- `LogAPrioriProbability()`

Read data

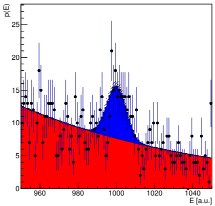
- text file
- ROOT tree
- anything in C++

Predefined model

template fit: signal + bkg

```
// define the model
BCMTF m("SingleChannelMTF");
m.AddChannel("channel1");
m.SetData("channel1", hist_data);
m.AddProcess("background", 200., 400.);
m.SetTemplate("channel1", "background",
      hist_background, 1.0);
m.SetPriorGauss("background", 300., 10.);
m.AddProcess("signal", 0., 200.);
m.SetTemplate("channel1", "signal", hist_signal, 1.0);
m.SetPriorConstant("signal");
```

```
// analyze model
m.MarginalizeAll();
m.PrintAllMarginalized("marg.pdf");
m.FindMode(m.GetBestFitParameters());
m.PrintStack(0, m.GetBestFitParameters(), "stack.pdf")
```

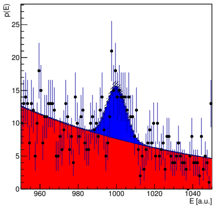


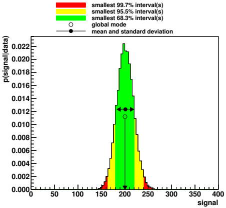
PREDEFINED MODEL

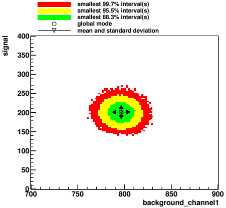
template fit: signal + bkg

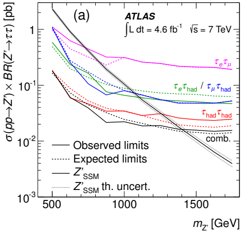
```
// define the model
BCMTF m("SingleChannelMTF");
m.AddChannel("channel1");
m.SetData("channel1", hist_data);
m.AddProcess("background", 200., 400.);
m.SetTemplate("channel1", "background",
      hist_background, 1.0);
m.SetPriorGauss("background", 300., 10.);
m.AddProcess("signal", 0., 200.);
m.SetTemplate("channel1", "signal", hist_signal, 1.0);
m.SetPriorConstant("signal");
```

```
// analyze model
m.MarginalizeAll();
m.PrintAllMarginalized("marg.pdf");
m.FindMode(m.GetBestFitParameters());
m.PrintStack(0, m.GetBestFitParameters(), "stack.pdf")
```



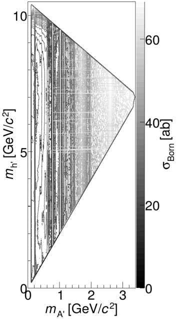






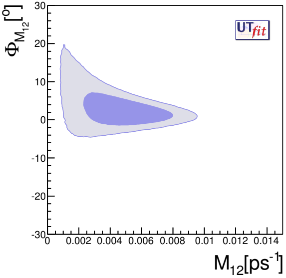
ATLAS: Z' search

Phys. Lett. B 719 (2013)



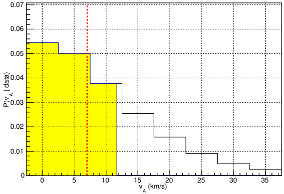
Belle: dark-photon search

arXiv:1502.00084



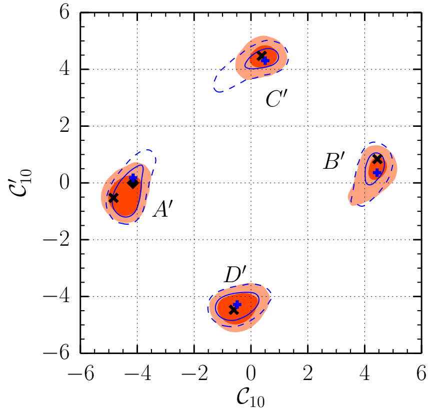
UTFIT: D meson mixing

arXiv:1402.1664

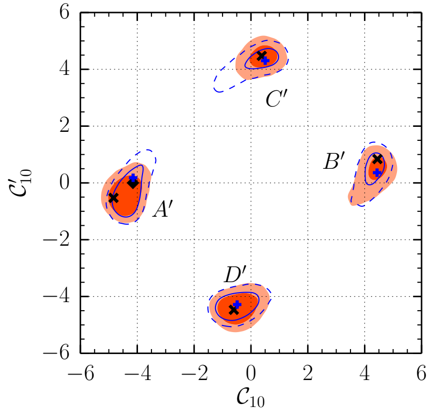


PAMELA: cosmic-ray proton spectrum

arXiv:1306.1354



see C. Bobeth's talk



see C. Bobeth's talk

Challenges

- high dimensional: (30-40)D
- isolated modes
- slow likelihood $\mathcal{O}(1s)$

$\Rightarrow$ Metropolis Hastings not enough

Sampling algorithm

Importance sampling

$$\int P = \int \frac{P}{q} q \approx \frac{1}{N} \sum_i \frac{P(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)},\ \boldsymbol{\theta} \sim q$$

SAMPLING ALGORITHM

IMPORTANCE SAMPLING

$$\int P = \int \frac{P}{q} q \approx \frac{1}{N} \sum_i \frac{P(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}, \; \boldsymbol{\theta} \sim q$$

MIXTURE PROPOSAL

$$q(\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Sampling algorithm

Importance sampling

$$\int P = \int \frac{P}{q} q \approx \frac{1}{N} \sum_i \frac{P(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}, \ \boldsymbol{\theta} \sim q$$

Mixture proposal

$$q(\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Advantages

- evidence + marginals
- massively parallel
- multiple modes

Sampling algorithm

Importance sampling

$$\int P = \int \frac{P}{q} q \approx \frac{1}{N} \sum_i \frac{P(\boldsymbol{\theta}_i)}{q(\boldsymbol{\theta}_i)}, \; \boldsymbol{\theta} \sim q$$

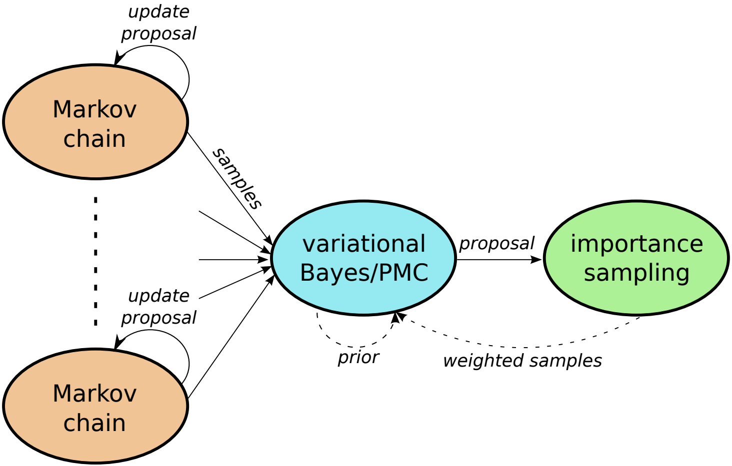
Mixture proposal

$$q(\boldsymbol{\theta}) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(\boldsymbol{\theta} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Advantages

- evidence + marginals
- massively parallel
- multiple modes

$\Rightarrow$ how to adjust proposal q?



hierarchical clustering + PMC: arXiv:1304.7808

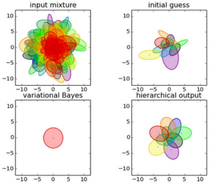
variational Bayes: S. Jahn's master's thesis (TU Munich, Feb 2015) pypmc 1.0 documentation »

pypmc

pypmc is a python package focusing on adaptive importance sampling. It can be used for integration and sampling from a user-defined target density. A typical application is Bayesian inference, where one wants to sample from the posterior to marginalize over parameters and to compute the evidence. The key idea is to create a good proposal density by adapting a mixture of Gaussian or student's t components to the target density. The package is able to efficiently integrate multimodal functions in up to about 30-40 dimensions at the level of 1% accuracy or less. For many problems, this is achieved without requiring any manual input from the user about details of the function. Importance sampling supports parallelization on multiple machines via mpi4py.

Useful tools that can be used stand-alone include:

- importance sampling (sampling & integration)
- adaptive Markov chain Monte Carlo (sampling)
- variational Bayes (clustering)



4.6. MCMC + variational Bayes

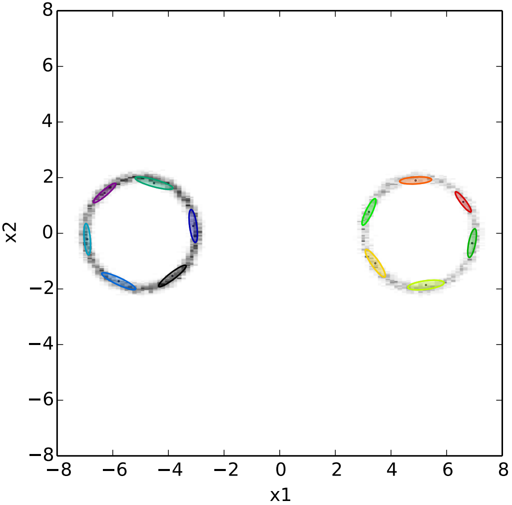
```
'''This example illustrates how pypmc can be used to int
non-negative function. The presented algorithm needs ver
analytical knowledge about the function.

'''

from __future__ import print_function
import numpy as np
import pypmc
```

MAIN FEATURES

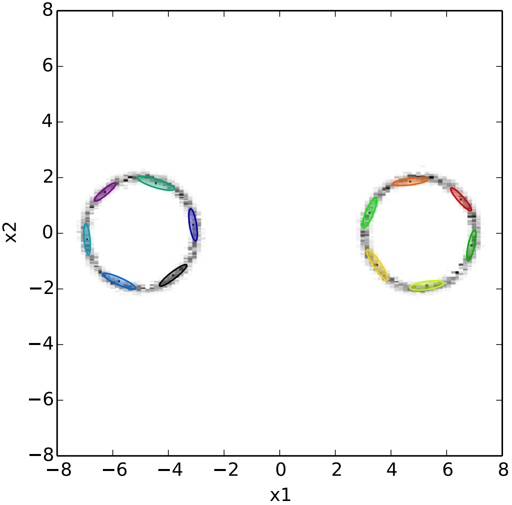
- Markov chains
- importance sampling (with `MPI`)
- variational Bayes (GMM)
- population Monte Carlo (Gauss + Student's t)

ORDER NOW

```
pip install pypmc
```



MCMC + variational Bayes (VB)



(MCMC + VB) + (IS + VB)

Improvements under development

- factorized priors $P(\boldsymbol{\theta}|M) = \prod_i P(\theta_i|M)$
  $\Rightarrow$ community extensible
- sharing samples as ROOT files (even w/o the model)
  $\Rightarrow$ uncertainty propagation, replotting
- multivariate proposal $\Rightarrow$ big speed-up in high dimensions
- evidence from MCMC arXiv:1410.7149

$\Rightarrow$ release in summer 2015

BAT 2.0

Wishlist for the future

- performance: threads + MPI for tough problems
- interface from C++ to python, mathematica ...
- sampling algorithms: MCMC, Hamiltonian MC, nested sampling, variational Bayes + importance sampling ...

$\Rightarrow$ BAT 2.0 ready for Belle II analyses

SUMMARY

1. Bayes: random numbers
2. BAT well usable for small-medium scale problems
3. many more powerful sampling algorithms in BAT 2.0