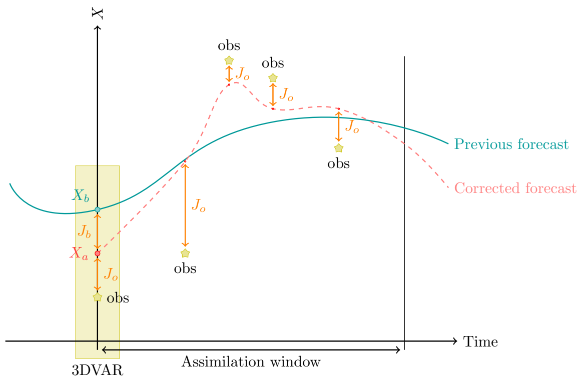
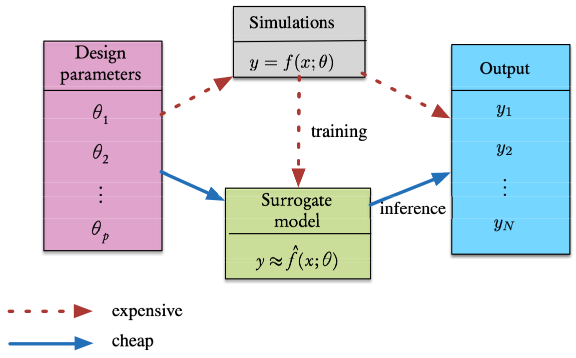
DA and SciML Advanced Course Projects

Mark Asch - CSU/IMU/VLP, Philippines - 2023





Project Guidelines

- The final objective of the course: each participant prepares a code project that implements one or more of the scientific machine learning concepts that we have learned, including the issue of ethics.

- This project can take one of the following forms:

  ⇒ extending one or more of the examples, or broadening their application

  ⇒ attempting to reproduce some known results from a research project or paper

  ⇒ trying to apply SciML to a new research topic that has not yet been considered

- During the afternoon workshops, we will discuss the potential topics and guide you in your choice.

- Each project will be presented and dicussed in the final two workshop sessions.

- A jury will award prizes (in the form of textbooks, kindly donated by SIAM) to the best projects.

PROJECT PROPOSALS

EnKF for Epidemiology

- We can use Kalman Filters in epidemiological studies, for both state and parameter estimation.

- Two papers study this approach:

  ⇒ Engbert, at al. Sequential Data Assimilation of the Stochastic SEIR Epidemic Model for Regional COVID-19 Dynamics. *Bulletin of Mathematical Biology* (2021) 83:1

  ⇒ Lal, et al. An application of the ensemble Kalman filter in epidemiological modelling. *Plos One* 16(8): e0256227. (2023)

- Two possible projects:

  1. Read, understand and reproduce some of their results.
  2. Read, understand and use their approach to solve your own epidemiological problem.

Disease Diagnosis

- Read the short paper `[Poudel2022]DiseaseDiag.pdf`
- Reproduce the results on the PIMA Indian diabetes data.
- Apply the methods to some other disease diagnosis problem.

PINN

- Use PINN to solve a partial differential equation that describes an interesting context in your research
- Use PINO to solve a partial differential equation that describes an interesting context in your research
- Compare the two.

DINN

Disease informed NNs - see paper of Shaier, Raissi and Seshaiyer [Raissi2022]DiNN.pdf

- read and understand the paper
- use their PINN approach to solve a basic SIR model
- apply the DINN approach to a more realistic epidemiological problem, either from the paper, or from other data sources.

SUMO

- For a dataset that describes some complex system

  $\Rightarrow$ perform extensive EDA

  $\Rightarrow$ define suitable features and response variables

  $\Rightarrow$ develop a SUMO based on the simplest possible ML methods

Probabilistic Programming using `pyro`

- learn pyro `https://docs.pyro.ai/en/stable/getting_started.html`
- use pyro to solve epidemiological models

  ⇒ `https://pyro.ai/examples/epi_intro.html`

  ⇒ any other problem in epidemiology

PROBABILISTIC PROGRAMING

Probabilistic Programming - introduction, examples and references

- Probabilistic programming (PP) attempts to provide a framework, and the corresponding programming tools, that facilitate design in presence of uncertainties.

- The approach is invariably Bayesian and will thus be more or less compute-intensive. This should not be considered an obstacle, since the potential benefits are huge, especially in the optic of decision-making and risk evaluation. In summary:

- PP provides a framework for building AI systems that reason about uncertainty and learn from data. Combined with decision theory, where the optimization part comes in, it forms the foundation of rational decision-making systems.

- PP provides a solid foundation for data-driven science, in general. It provides an open language for sharing models and enabling reproducibility.

- PP, through Bayes Law, can be used for

⇒ Learning:

$$\mathrm{P}(\theta \mid \mathcal{D}, m) = \frac{\mathrm{P}(\mathcal{D} \mid \theta, m)\mathrm{P}(\theta \mid m)}{\mathrm{P}(\mathcal{D} \mid m)},$$

where $\mathrm{P}(\mathcal{D} \mid \theta, m)$ is the likelihood of parameters θ in model m, the prior probability of θ is $\mathrm{P}(\theta \mid m)$ and $\mathrm{P}(\theta \mid \mathcal{D}, m)$ is the posterior probability of the sought for parameters.

⇒ Prediction:

$$\mathrm{P}(x \mid \mathcal{D}, m) = \int \mathrm{P}(x \mid \theta, \mathcal{D}, m)\mathrm{P}(\theta \mid \mathcal{D}, m)\, \mathrm{d}\theta.$$

⇒ Model selection:

$$\mathrm{P}(m \mid \mathcal{D}) = \frac{\mathrm{P}(\mathcal{D} \mid m)\mathrm{P}(m)}{\mathrm{P}(\mathcal{D})}.$$

Probabilistic Programming - languages

Numerous languages have been developed for PP. Among these are:

- BUGS[1] (circa 1990), a declarative, graphical language. This was the pioneer.
- STAN,[2] from the Bayesian MCMC community, headed by A. Gelman (circa 2015). Widely used in the broad academic community, including social and biomedical sciences.
- Pyro,[3] a recent framework, built upon PyTorch . Used in ML/AI industry and intrinsically linked to the deep learning capabilities of PyTorch. The web-

[1] `https://www.mrc-bsu.cam.ac.uk/software/bugs/`
[2] `https://mc-stan.org/`
[3] `https://pyro.ai/`

site provides a vast array of worked examples ranging from simple linear regression to deep neural networks. It also has Bayesian Optimization capabilities—see `http://pyro.ai/examples/bo.html`.

- Turing[4] (more recent, based on Julia). More academic than Pyro, for example. Being based on Julia gives it a cutting edge in high performance, parallel and GPU-based computing.
- PyMC3,[5] a Python library for Bayesian modeling and Probabilistic Machine Learning (PML), leveraging the Theano library.

[4]`https://turing.ml/dev/`
[5]`https://docs.pymc.io/`

Probabilistic Programming - structure

Although Bayes' formula and the resulting Bayesian statistics are conceptually simple, we have noted that fully probabilistic models often lead to analytically and computationally intractable expressions. For many years, this constituted a serious barrier, limiting and hindering the wide adoption of Bayesian methods. Fortunately, we now have numerical methods and high performance computing capabilities that are capable, at least in principle, of solving any inference problem.

The question of automating the inference process has led to the development of probabilistic programming languages (PPLs). These provide a clear separation between model definition and inference. They are—by design—relatively simple, intuitive, and possess easy to read syntax that closely imitates the actual statistical syntax used to describe probabilistic models.

PPL models are defined inside a context manager. To add a probability distribution to a model, requires only a single line of code. Distributions can be combined and can be used as priors, in the case of unobserved variables, or likelihoods in the case of observed variables. If we pass data to a distribution, it becomes a likelihood. Then, sampling can be performed with a single line as well, to produce a posterior. Finally, one can readily get samples from the posterior distribution, which will be a faithful, probabilistic representation of the logical consequences of our model and our data.

Probabilistic Programming - examples

We present examples that show how different problems can be solved using Probabilistic Programming. In these examples we use `pymc3`, but they can easily be executed with any of the other languages cited above.

Probabilistic Programming - Example 1

Example 1 (Coin-Tossing with PP)**.** We solve a simple coin-tossing posterior example, but without using a conjugate prior. Instead, we will use MCMC sampling based on the NUTS sampler in PyMC3. The results are shown in Figure 1.

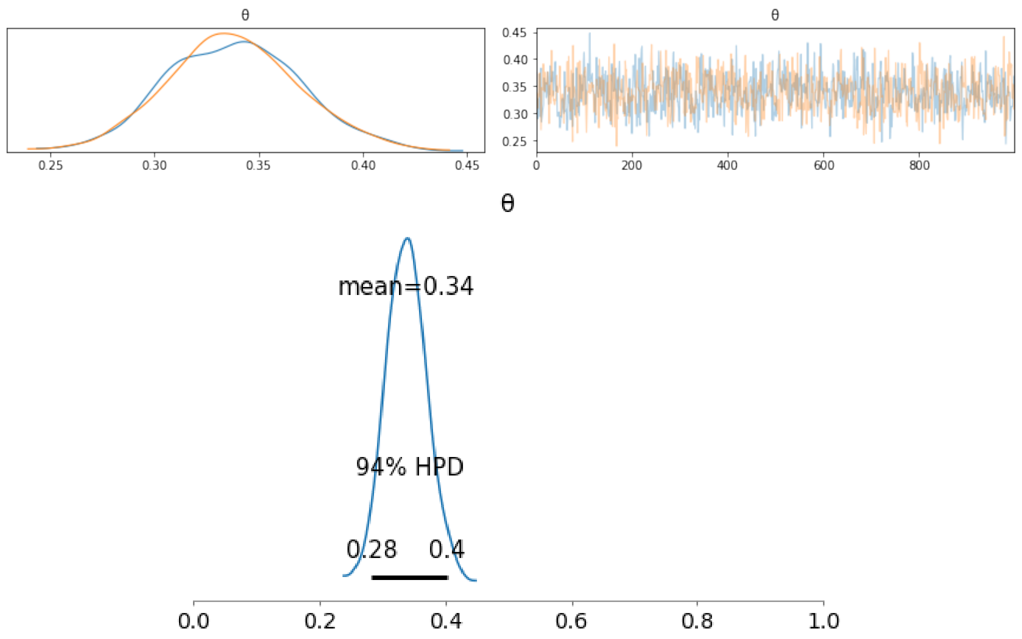


Figure 1: Computation of the posterior for a binomial experiment with uninformative prior. Topmost graphs show the sampling output from PyMC3. The bottom plot shows the posterior with a 94% HPD. Results computed by `BayesBernBetaMC.ipynb`.

```
#- BayesBernBetaMC.ipynb -#
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
import pymc3 as pm
import arviz as az
# initialize
np.random.seed(123)
trials = 200
thetatrue = 0.35 # unknown value in a real experi
data = stats.bernoulli.rvs(p=thetatrue, size=tria
# print outcomes
data
# define model and execute sampling
with pm.Model() as Bernbetamodel:
# prior is uninformative/uniform
theta = pm.Beta('\theta', alpha=1., beta=1.)
# likelihood
y = pm.Bernoulli('y', p=theta, observed=data)
# inference:
trace = pm.sample(1000, randomseed=123)
# plot the posterior
az.plottrace(trace)
az.summary(trace)
# plot posterior with 94% HPD
ax2, = az.plotposterior(trace)
ax2.setxlim(0, 1)
```

Probabilistic Programming - Example 2

Next, we consider a classical linear regression problem. Here we will use PP to solve it. This is an important example that brings out the similarities and differences, advantages and inconveniences of the three approaches: variational, Bayes, and PP based on MCMC.

Example 2 (Bayesian Linear Regression using PP). `https://github.com/markasch/DT-tbx-examples/blob/master/11uq/x11p24_bayes_lin_reg_pp.ipynb`

Probabilistic Programming - Example 3

Finally, we show how PP can be used for parameter estimation in differential equations—the Bayesian inverse problem.

Example 3 (Bayesian Inverse Problem for ODEs using PP)**.** We use the `pymc3` framework to solve parameter estimation, Bayesian inverse problems for some ODEs.

`https://github.com/markasch/DT-tbx-examples/blob/master/11uq/x11p25_bayes_ode_mcmc.ipynb`

References

1. https://en.wikipedia.org/wiki/Data_assimilation

2. G. Evensen. *Data assimilation, The Ensemble Kalman Filter*, 2nd ed., Springer, 2009.

3. E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability.* Cambridge University Press, 2003.

4. K. Law, A. Stuart, and K. Zygalakis. *Data Assimilation. A Mathematical Introduction.* Springer, 2015.

5. A. Tarantola. *Inverse problem theory and methods for model parameter estimation.* SIAM. 2005.

6. G. James, D. Witten, T. Hastie, R. Tibshirani. *An Introduction to Statistical Learning with Applications in R.* Springer. 2013.

http://www.statlearning.com
https://hastie.su.domains/ISLR2/ISLRv2_corrected_
June_2023.pdf

7. G. James, D. Witten, T. Hastie, R. Tibshirani. *An Introduction to Statistical Learning with Applications in Python.* Springer. 2023.
https://hastie.su.domains/ISLP/ISLP_website.pdf

8. Rachel Schutt and Cathy O'Neil. *Doing Data Science.* O'Reilly. 2014.

9. I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning.* MIT Press. 2016.

   http://www.deeplearningbook.org

10. M. Kuhn, K. Johnson. Applied Predictive Modeling. Springer 2018.
   http://appliedpredictivemodeling.com/

Software for DA

Various open-source repositories and codes are available for both academic and operational data assimilation.

1. DARC: `https://research.reading.ac.uk/met-darc/` from Reading, UK.

2. DAPPER: `https://github.com/nansencenter/DAPPER` from Nansen, Norway.

3. DART: `https://dart.ucar.edu/` from NCAR, US, specialized in ensemble DA.

4. OpenDA: `https://www.openda.org/`.

5. Verdandi: `http://verdandi.sourceforge.net/` from INRIA, France.

6. PyDA: `https://github.com/Shady-Ahmed/PyDA`, a Python implementation for academic use.

7. Filterpy: `https://github.com/rlabbe/filterpy`, dedicated to KF variants.

8. EnKF; `https://enkf.nersc.no/`, the original Ensemble KF from Geir Evensen.

Software for ML

1. R:
   https://cran.r-project.org/
2. scikit-learn:
   https://scikit-learn.org/stable/
3. PyTorch:
   https://pytorch.org/get-started/locally/