Course Logistics

Instructor	Prof. Daniel Sanz-Alonso	`sanzalonso@uchicago.edu`	Jones 113
Teaching Assistant	Nathan Waniorek	`waniorek@uchicago.edu`	
Website	`canvas.uchicago.edu`	(all grades, assignments, announcements, etc.)	

Mon	Tue	Wed	Thu	Fri
	Lecture: 9:30–10:50 Daniel OH 4pm		Lecture: 9.30-10:50	

By default, the instructor's office hours will be in-person at Jones 113, unless otherwise announced on Canvas.
Nathan's Office Hours (Jones 226): 12/3 and 12/4 5pm – 7.00pm.

Grading

- Your course grade: Three Computational Projects: 25% each. Presentation: 25%.
- As a default, students will receive a quality grade (A, A-, B+, ...). Alternatively, you may register for grade R (audit), or may request a P/F (pass/fail) grade or a W (withdrawal). Requests for P/F or W must be sent from your `uchicago.edu` email account, to the instructor, before November 30. A grade I (incomplete) will be given only in clear cases of emergency and must be approved by the department chair.
- Given that all the assignment deadlines are announced months in advance, late submissions will only be accepted in cases of serious health or family emergencies. If an emergency occurs, please let the instructor know by email as soon as possible.

Presentation

The goal is to read and explain clearly to the other students in the class a paper or collection of papers related to the class. The papers should all relate to one of the chapters in the lecture notes. You may focus on paper(s) discussed in the bibliography section or discuss other papers, as long as you clearly explain the connection to one of the chapters. Depending on the number of students, the presentations will be individual or by groups; we will provide more details, including the duration of the presentations, once we know the number of students in the class.

Projects

We recommend to use Python for the computational projects, since the provided models and libraries all use Python.
Resources

- The code for the ML and numerical models described below is available at
  `https://github.com/eviatarbach/acm270_projects`.
- For the data assimilation problems, the Python DAPPER library provides nice implementations of many algorithms, and may be used.
- The Python library PyMC provides implementations of Markov chain Monte Carlo (MCMC) algorithms.
- Probabilistic scoring rules, including the continuous ranked probability score, are implemented in the Python properscoring package.
- `jax` is a Python library that allows one to autodifferentiate code, as well perform just-in-time compilation (allowing it to run much faster than regular Python code). The cd_dynamax library provides implementations of many data assimilation algorithms in `jax`. For the projects below that involve autodifferentiation, use of `jax` is recommended.

1. **Inverse problems with ML surrogate models**

   Description

   This project will involve solving inverse problems using ML surrogate models. This is motivated by the fact that the latter are often faster to run than the true forward model. They can hence significantly speed-up multi-query methods such as MCMC.

   The two models to be used will be

   - The 40-dimensional Lorenz96 model, a chaotic ODE meant to represent the dynamics of the midlatitude atmosphere. Both a numerical integrator of the ODE and an ML model are provided in the `lorenz96` folder. The ML model is described in [1].
   - The Darcy equation, describing flow through a porous medium. Here, a Fourier neural operator surrogate [4] is provided in the `darcy` folder.

   Project goals

   - Solve an inverse problem in the numerical Lorenz96 model. Take the forward model to be an integration of the Lorenz96 dynamics for 0.2 time units[1] from some initial state, and try to recover this initial state from a noisy observation of the state at the end of the interval. Obtain samples of the posterior using MCMC. Do this for both full observations $H = I$ and partial observations (observing every other location).
   - Repeat the same with the ML Lorenz96 model and compare the results, using the surrogate likelihood for MCMC.
   - Solve the inverse problem of obtaining the permeability field given the pressure field for the Darcy equation. First, take a ground truth permeability field and add noise to the resulting pressure field (permeability–pressure pairs can be obtained as discussed below). Then, use the Darcy ML surrogate likelihood in MCMC. Compare the posterior you obtain with MCMC with the ground truth input.

   Tips

   - Look in `lorenz96/example.py` to see how to run the Lorenz96 numerical model and surrogate, and in `darcy/example.py` to see how to run the Darcy flow surrogate.
   - The "true" numerical model for the Darcy flow is not provided, but `darcy/example.py` shows how to load ground truth input–output pairs, which you will use to generate the observations for the inverse problem.

   Bonus

   Implement a derivative-based MCMC sampler such as Hamiltonian Monte Carlo as opposed to random walk Metropolis.

2. **Ensemble data assimilation with ML model**

   Description

   This project will consist of implementing the ensemble Kalman filter using an ML forecast model. In particular, since the ML forecast is significantly less computationally expensive than the numerical one, large ensembles of the ML forecasts can be generated.

   The project will make use of a two-layer quasigeostrophic model, a simplified form of the equations governing atmospheric flow. In the `qg` folder, we have made available both a numerical solver and an ML surrogate, which uses a U-Net architecture described in [2].

   Project goals

   - Produce a true trajectory and synthetic observations using the numerical model.
   - Perform DA by applying an ensemble Kalman filter to the ML model.

[1] 0.2 time units leads to a non-trivial inverse problem but is still well under the mixing time.

- Compare the filtering performance using the ML model to filtering using the numerical model. The performance should be measured using by verifying against the true trajectory using one of the probabilistic scoring rules discussed in class (e.g., the continuous ranked probability score). Compare the performance using different ensemble sizes; in particular, using a larger ML ensemble than numerical ensemble.

Tips

- Look in `qg/example.py` to see how to run the numerical and ML models.
- It may be a good idea to test the filter with a simpler model first, such as Lorenz96 (also provided; see project 1).
- The models are configured such that each evaluation of the function corresponds to a forecast 1.0 time units into the future, which corresponds to about 6 hours on Earth's atmosphere. This is a good choice for the assimilation interval.
- Unless very large ensembles are used, localization will be required to have a stable filter.

Bonus Perform data assimilation using both a physical ensemble and an ML ensemble using one of the multi-model or multifidelity methods described in the lecture notes.

3. **Model error correction in ensemble data assimilation**

Description

This project focuses on learning a model error correction in data assimilation.

Here we will take the true system to be the two-scale Lorenz96 model, and the imperfect model to be the single-scale Lorenz96. The single-scale Lorenz96 will incur error due to the lack of the small-scale interactions.

Main project goals (80%)

- Produce a true trajectory and noisy observations using the two-scale Lorenz96.
- Assimilate those observations into the single-scale Lorenz96 using an ensemble Kalman filter.
- Train a neural network (your choice of architecture) to predict the analysis increment (analysis − forecast) from the forecast state.
- Run the EnKF again with the imperfect model, but add the neural network correction. Compare the filtering performance to that of the uncorrected model using one of the probabilistic scoring rules discussed in class.

Additional project goals (20%)

Instead of using analysis increments to learn the model error, use the autodifferentiable EnKF [3]. This will require jax. Compare the results to the correction learned by analysis increments.

Tips

- The single-scale and two-scale Lorenz96 models are provided in `lorenz96/numerical_model.py`.
- Due to the model error, assimilating observations of the two-scale Lorenz96 model into the single-scale version will require inflation to function well.
- The time-step 0.005 will work for integrating the two-scale Lorenz96 model with 4th-order Runge–Kutta.

Collaboration Policy

You are encouraged to discuss the assignments with other students, but you must write up solutions on your own. Copying from another student's solution, or sharing your own solutions with another student or posting them online, are all considered violations of this policy. Please ask the instructor if you have any concerns or questions regarding the expectations for collaboration and academic honesty.

Course Contents

The course will cover selected topics on machine learning techniques for inverse problems and data assimilation. Lecture notes will be provided.

PART I: Inverse Problems

1. Bayesian Inversion.
2. Variational Inference.
3. Forward Surrogate Model.
4. Learning Prior and Regularizers.
5. Transporting to the Posterior.
6. Learning Dependence on Data.

PART II: Data Assimilation

1. Filtering and Smoothing.
2. Learning Forecast and Observation Models.
3. Learning Parameterized Filters and Smoothers.
4. Learning Filters and Smoothers Using Transport.
5. Data Assimilation Using Learned Forecasting Models.

PART III: Learning Frameworks

1. Metrics, Divergences, and Scoring Rules.
2. Unsupervised Learning and Generative Modeling.
3. Supervised Learning.

***Tentative* Course Schedule**

Week	Date	Material	Due
1	10/1	PART III (Ch 12)	
	10/3	PART III (Ch 12)	
2	10/8	PART III (Ch 12)	
	10/10	PART III (Ch 12)	
3	10/15	PART III (Ch 13)	
	10/17	PART III (Ch 13)	
4	10/22	PART III (Ch 13 and 14)	
	10/24	PART III (Ch 14)	
5	10/29	PART I (Ch 1 and 2)	
	10/31	PART I (Ch 3 and 4)	
6	11/5	PART I (Ch 5 and 6)	
	11/7	PART II (Ch 7 and 8)	
7	11/12	PART II (Ch 9 and 10)	
	11/14	PART II (Ch 11)	
8	11/19	Project and Presentation Consultation, Other Topics	
	11/21	Project and Presentation Consultation, Other Topics	
	11/26	Thanksgiving week—no class	
	11/28	Thanksgiving week—no class	
9	12/3	Presentations	
	12/5	Presentations	PROJECTS DUE

References

[1] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model. page 101171.

[2] Ashesh Chattopadhyay, Ebrahim Nabizadeh, Eviatar Bach, and Pedram Hassanzadeh. Deep learning-enhanced ensemble-based data assimilation for high-dimensional nonlinear dynamical systems. 477:111918.

[3] Yuming Chen, Daniel Sanz-Alonso, and Rebecca Willett. Auto-differentiable Ensemble Kalman Filters.

[4] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations.