Machine Learning in Data Assimilation

Eviatar Bach

13 June 2025

Inverse Problems and Data Assimilation: A Machine Learning Approach

Eviatar Bach ♯♮, Ricardo Baptista ♯, Daniel Sanz-Alonso ♭, Andrew Stuart ♯

https://www.arxiv.org/abs/2410.10523

Preliminaries

Data assimilation

Consider the dynamical system

$$x_{j+1}^{\mathrm{t}} = \mathcal{M}(x_j^{\mathrm{t}}) + \xi_j^{\dagger}, \tag{1a}$$

$$x_0^{\mathrm{t}} \sim \mathcal{N}(m_0, C_0), \quad \xi_j^{\dagger} \sim \mathcal{N}(0, Q) \text{ i.i.d.}, \tag{1b}$$

Data assimilation

Consider the dynamical system

$$x_{j+1}^{\dagger} = \mathcal{M}(x_j^{\dagger}) + \xi_j^{\dagger}, \tag{1a}$$

$$x_0^{\dagger} \sim \mathcal{N}(m_0, C_0), \quad \xi_j^{\dagger} \sim \mathcal{N}(0, Q) \text{ i.i.d.}, \tag{1b}$$

The observations are given by

$$y_{j+1}^{\dagger} = h(x_{j+1}^{\dagger}) + \eta_{j+1}^{\dagger}, \tag{2a}$$

$$\eta_j^{\dagger} \sim \mathcal{N}(0, R) \quad \text{i.i.d.} \tag{2b}$$

Data assimilation

Consider the dynamical system

$$x_{j+1}^{\dagger} = \mathcal{M}(x_j^{\dagger}) + \xi_j^{\dagger}, \tag{1a}$$

$$x_0^{\dagger} \sim \mathcal{N}(m_0, C_0), \quad \xi_j^{\dagger} \sim \mathcal{N}(0, Q) \text{ i.i.d.}, \tag{1b}$$

The observations are given by

$$y_{j+1}^{\dagger} = h(x_{j+1}^{\dagger}) + \eta_{j+1}^{\dagger}, \tag{2a}$$

$$\eta_j^{\dagger} \sim \mathcal{N}(0, R) \quad \text{i.i.d.} \tag{2b}$$

Data assimilation

We define, for a fixed integer J,

$$X^{\mathrm{t}} = \{x_0^{\mathrm{t}}, \ldots, x_J^{\mathrm{t}}\},\ Y^{\dagger} = \{y_1^{\dagger}, \ldots, y_J^{\dagger}\},\ Y_j^{\dagger} = \{y_1^{\dagger}, \ldots, y_j^{\dagger}\}.$$

Data assimilation

We define, for a fixed integer J,

$$X^{\mathrm{t}} = \{x_0^{\mathrm{t}}, \ldots, x_J^{\mathrm{t}}\},\ Y^{\dagger} = \{y_1^{\dagger}, \ldots, y_J^{\dagger}\},\ Y_j^{\dagger} = \{y_1^{\dagger}, \ldots, y_j^{\dagger}\}.$$

Three distinct tasks in data assimilation (DA):

- The problem of *smoothing* is to obtain the conditional distribution $\mathbb{P}(X^{\mathrm{t}}|Y^{\dagger})$ (e.g., ensemble Kalman smoother).

Data assimilation

We define, for a fixed integer J,

$$X^{\mathrm{t}} = \{x_0^{\mathrm{t}}, \ldots, x_J^{\mathrm{t}}\},\ Y^{\dagger} = \{y_1^{\dagger}, \ldots, y_J^{\dagger}\},\ Y_j^{\dagger} = \{y_1^{\dagger}, \ldots, y_j^{\dagger}\}.$$

Three distinct tasks in data assimilation (DA):

- The problem of *smoothing* is to obtain the conditional distribution $\mathbb{P}(X^{\mathrm{t}}|Y^{\dagger})$ (e.g., ensemble Kalman smoother).
- The problem of *filtering* is to obtain $\mathbb{P}(x_j^{\dagger}|Y_j^{\dagger})$ for any $j \leq J$ (e.g., ensemble Kalman filter).

Data assimilation

We define, for a fixed integer J,

$$X^{\mathrm{t}} = \{x_0^{\mathrm{t}}, \ldots, x_J^{\mathrm{t}}\},\ Y^{\dagger} = \{y_1^{\dagger}, \ldots, y_J^{\dagger}\},\ Y_j^{\dagger} = \{y_1^{\dagger}, \ldots, y_j^{\dagger}\}.$$

Three distinct tasks in data assimilation (DA):

- The problem of *smoothing* is to obtain the conditional distribution $\mathbb{P}(X^{\mathrm{t}}|Y^{\dagger})$ (e.g., ensemble Kalman smoother).
- The problem of *filtering* is to obtain $\mathbb{P}(x_j^{\dagger}|Y_j^{\dagger})$ for any $j \leq J$ (e.g., ensemble Kalman filter).
- The problem of *state estimation* is to obtain a $x_j \approx x_j^{\mathrm{t}}$ in some norm (e.g., 3DVar).

ML in DA

ML in DA

How to incorporate machine learning (ML) into DA?

- Learning forecast model, learning corrections

ML in DA

How to incorporate machine learning (ML) into DA?

- Learning forecast model, learning corrections
- Using learned forecast model

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators
- End-to-end forecasting and DA using ML, e.g., score-based DA (Rozet and Louppe 2023), DA networks (Boudier et al. 2023)

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators
- End-to-end forecasting and DA using ML, e.g., score-based DA (Rozet and Louppe 2023), DA networks (Boudier et al. 2023)
- Learning smoothers

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators
- End-to-end forecasting and DA using ML, e.g., score-based DA (Rozet and Louppe 2023), DA networks (Boudier et al. 2023)
- Learning smoothers
- Learning observation operators and observation error covariances (Waller, Dance, and Nichols 2016)

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators
- End-to-end forecasting and DA using ML, e.g., score-based DA (Rozet and Louppe 2023), DA networks (Boudier et al. 2023)
- Learning smoothers
- Learning observation operators and observation error covariances (Waller, Dance, and Nichols 2016)
- Transport-based (Spantini, Baptista, and Marzouk 2022) and reinforcement learning-based (Hammoud et al. 2024) approaches to DA

ML in DA

Other topics (won't discuss):

- Learning filters and state estimators
- End-to-end forecasting and DA using ML, e.g., score-based DA (Rozet and Louppe 2023), DA networks (Boudier et al. 2023)
- Learning smoothers
- Learning observation operators and observation error covariances (Waller, Dance, and Nichols 2016)
- Transport-based (Spantini, Baptista, and Marzouk 2022) and reinforcement learning-based (Hammoud et al. 2024) approaches to DA
- Practical considerations

Automatic differentiation

Gradients are often difficult to obtain in closed form for complex cost functions.

Automatic differentiation

Gradients are often difficult to obtain in closed form for complex cost functions.

Finite difference approximations are often inaccurate and expensive for high-dimensional problems.

Automatic differentiation

Gradients are often difficult to obtain in closed form for complex cost functions.

Finite difference approximations are often inaccurate and expensive for high-dimensional problems.

Automatic differentiation (autodiff) involves repeated application of chain rule on elementary operations that enables computing derivatives accurately to working precision.

Automatic differentiation

Gradients are often difficult to obtain in closed form for complex cost functions.

Finite difference approximations are often inaccurate and expensive for high-dimensional problems.

Automatic differentiation (autodiff) involves repeated application of chain rule on elementary operations that enables computing derivatives accurately to working precision.

This can allow differentiation through model states (adjoint), model parameters, and DA algorithms, and enable use of gradient-based optimization.

Automatic differentiation

Suppose $y = f(x)$, where $y \in \mathbb{R}^{d_n}$ and $x \in \mathbb{R}^{d_0}$, and the derivative of f is not readily available in closed-form.

Automatic differentiation

Suppose $y = f(x)$, where $y \in \mathbb{R}^{d_n}$ and $x \in \mathbb{R}^{d_0}$, and the derivative of f is not readily available in closed-form.

We assume the implementation of f in computer code is made up of n elementary operations $f_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ (for instance, addition, multiplication, logarithms, etc.), such that

$$f(x) = f_n \circ f_{n-1} \circ \cdots \circ f_1(x), \tag{3}$$

where the Jacobians for these elementary operations, $Df_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i \times d_{i-1}}$, are available in closed form.

Automatic differentiation

Suppose $y = f(x)$, where $y \in \mathbb{R}^{d_n}$ and $x \in \mathbb{R}^{d_0}$, and the derivative of f is not readily available in closed-form.

We assume the implementation of f in computer code is made up of n elementary operations $f_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$ (for instance, addition, multiplication, logarithms, etc.), such that

$$f(x) = f_n \circ f_{n-1} \circ \cdots \circ f_1(x), \tag{3}$$

where the Jacobians for these elementary operations, $Df_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i \times d_{i-1}}$, are available in closed form.

Writing the partial evaluations up to $i \leq n$ as

$$g_i = (f_i \circ \cdots \circ f_1)(x),$$

by the chain rule we have that

$$D_x f(x) = (D_{g_{n-1}} f_n(g_{n-1})) \cdots (D_{g_1} f_2(g_1))(D_x f_1(x)). \tag{4}$$

Automatic differentiation

This suggests the following algorithm (*forward mode* automatic differentiation):

1: **Input**: The functions $\{f_i(\cdot)\}_{i=1}^n$, their corresponding Jacobians $\{Df_i(\cdot)\}_{i=1}^n$, and the function input x.
2: Set $g_1 = f_1(x)$ and $J_1 = Df_1(x)$.
3: For $i = 2, \ldots, n$: set $g_i = f_i(g_{i-1})$ and $J_i = (Df_i(g_{i-1}))J_{i-1}$.
4: **Output**: The function output $y = f(x) = g_n$ and the derivative $Df(x) = J_n$.

Automatic differentiation

This suggests the following algorithm (*forward mode* automatic differentiation):

1: **Input**: The functions $\{f_i(\cdot)\}_{i=1}^n$, their corresponding Jacobians $\{Df_i(\cdot)\}_{i=1}^n$, and the function input x.
2: Set $g_1 = f_1(x)$ and $J_1 = Df_1(x)$.
3: For $i = 2, \ldots, n$: set $g_i = f_i(g_{i-1})$ and $J_i = (Df_i(g_{i-1}))J_{i-1}$.
4: **Output**: The function output $y = f(x) = g_n$ and the derivative $Df(x) = J_n$.

Reverse mode autodiff requires a backwards pass to compute the derivative, and is more efficient when $d_0 \gg d_n$ (many inputs), whereas forward mode is more efficient when $d_n \gg d_0$ (many outputs).

Learning forecast model

Learning forecast model

The dynamical model has parameters ϑ:

$$x_{j+1}^{\mathsf{t}} = \mathcal{M}_{\vartheta}(x_j^{\mathsf{t}}) + \xi_j^{\dagger}, \tag{5a}$$

$$x_0^{\mathsf{t}} \sim \mathcal{N}(m_0, C_0),\ \xi_j^{\dagger} \sim \mathcal{N}(0, Q(\vartheta)) \text{ i.i.d.} \tag{5b}$$

Learning forecast model

The dynamical model has parameters ϑ:

$$x_{j+1}^{\mathsf{t}} = \mathcal{M}_{\vartheta}(x_j^{\mathsf{t}}) + \xi_j^{\dagger}, \tag{5a}$$

$$x_0^{\mathsf{t}} \sim \mathcal{N}(m_0, C_0),\ \xi_j^{\dagger} \sim \mathcal{N}(0, Q(\vartheta)) \text{ i.i.d.} \tag{5b}$$

Example 1 (parameterized dynamics): $\mathcal{M} = \mathcal{M}_{\vartheta^{\dagger}}$ is parameterized, but the true parameter $\vartheta^{\dagger}$ is unknown and needs to be estimated.

Learning forecast model

The dynamical model has parameters ϑ:

$$x_{j+1}^{\mathrm{t}} = \mathcal{M}_\vartheta(x_j^{\mathrm{t}}) + \xi_j^\dagger, \tag{5a}$$

$$x_0^{\mathrm{t}} \sim \mathcal{N}(m_0, C_0),\ \xi_j^\dagger \sim \mathcal{N}(0, Q(\vartheta)) \text{ i.i.d.} \tag{5b}$$

Example 1 (parameterized dynamics): $\mathcal{M} = \mathcal{M}_{\vartheta^\dagger}$ is parameterized, but the true parameter $\vartheta^\dagger$ is unknown and needs to be estimated.

Example 2 (fully unknown dynamics): $\mathcal{M}$ is fully unknown and ϑ can represent, for example, the parameters of a neural network or Gaussian process surrogate model $\mathcal{M}_\vartheta$ for $\mathcal{M}$.

Learning forecast model

The dynamical model has parameters ϑ:

$$x_{j+1}^{\mathrm{t}} = \mathcal{M}_\vartheta(x_j^{\mathrm{t}}) + \xi_j^\dagger, \tag{5a}$$

$$x_0^{\mathrm{t}} \sim \mathcal{N}(m_0, C_0),\ \xi_j^\dagger \sim \mathcal{N}(0, Q(\vartheta)) \text{ i.i.d.} \tag{5b}$$

Example 1 (parameterized dynamics): $\mathcal{M} = \mathcal{M}_{\vartheta^\dagger}$ is parameterized, but the true parameter $\vartheta^\dagger$ is unknown and needs to be estimated.

Example 2 (fully unknown dynamics): $\mathcal{M}$ is fully unknown and ϑ can represent, for example, the parameters of a neural network or Gaussian process surrogate model $\mathcal{M}_\vartheta$ for $\mathcal{M}$.

Example 3 (model correction): $\mathcal{M}$ is unknown, but we have access to an inaccurate model $\mathcal{M}^{\text{approx}} \approx \mathcal{M}$. The goal is to learn ϑ so that $\mathcal{M}_\vartheta = \mathcal{M}^{\text{approx}} + \mathcal{M}_\vartheta^{\text{correction}}$ approximates $\mathcal{M}$ accurately.

Learning forecast model

We consider learning the parameters from partial, noisy observations.

Learning forecast model

We consider learning the parameters from partial, noisy observations.

If we have the true (or good enough) model and want to learn an ML surrogate, we can just use supervised learning.

Learning forecast model

We consider learning the parameters from partial, noisy observations.

If we have the true (or good enough) model and want to learn an ML surrogate, we can just use supervised learning.

We would like to learn ϑ by maximizing the likelihood

$$\mathbb{P}(Y^\dagger|\vartheta) = \int \mathbb{P}(Y^\dagger, X|\vartheta)\, dX. \tag{6}$$

Here the states X are unobserved latent variables.

Learning forecast model

Evaluating the preceding integral is intractable. Instead, we use *expectation–maximization* (EM) algorithm for maximum likelihood estimation.

Learning forecast model

Evaluating the preceding integral is intractable. Instead, we use *expectation–maximization* (EM) algorithm for maximum likelihood estimation.

EM consists of two steps:

1. *Expectation:* Take the expectation of the log-likelihood of the state and observations given the parameters ϑ_ℓ:

$$\mathrm{J}_\ell(\vartheta) = \mathbb{E}^{X \sim \mathbb{P}(X|Y^\dagger, \vartheta_\ell)}[\log \mathbb{P}(X, Y^\dagger|\vartheta)] \tag{7}$$

Learning forecast model

Evaluating the preceding integral is intractable. Instead, we use *expectation–maximization* (EM) algorithm for maximum likelihood estimation.

EM consists of two steps:

1. *Expectation:* Take the expectation of the log-likelihood of the state and observations given the parameters ϑ_ℓ:

$$\mathrm{J}_\ell(\vartheta) = \mathbb{E}^{X \sim \mathbb{P}(X|Y^\dagger, \vartheta_\ell)}[\log \mathbb{P}(X, Y^\dagger|\vartheta)] \tag{7}$$

2. *Maximization:*

$$\vartheta_{\ell+1} = \arg\max_{\vartheta} \mathrm{J}_\ell(\vartheta) \tag{8}$$

Learning forecast model

Evaluating the preceding integral is intractable. Instead, we use *expectation–maximization* (EM) algorithm for maximum likelihood estimation.

EM consists of two steps:

1. *Expectation:* Take the expectation of the log-likelihood of the state and observations given the parameters ϑ_ℓ:

$$\mathrm{J}_\ell(\vartheta) = \mathbb{E}^{X\sim\mathbb{P}(X|Y^\dagger,\vartheta_\ell)}[\log \mathbb{P}(X, Y^\dagger|\vartheta)] \tag{7}$$

2. *Maximization:*

$$\vartheta_{\ell+1} = \arg\max_{\vartheta} \mathrm{J}_\ell(\vartheta) \tag{8}$$

This algorithm has the property that

$$\mathbb{P}(Y^\dagger|\vartheta_{\ell+1}) \geq \mathbb{P}(Y^\dagger|\vartheta_\ell). \tag{9}$$

Learning model from reanalysis

Suppose we have a reanalysis

$$\{x_j^{\mathrm{a}}\}_{j=1}^{J}.$$

Learning model from reanalysis

Suppose we have a reanalysis

$$\{x_j^{\mathrm{a}}\}_{j=1}^{J}.$$

We can train an ML model to predict

$$x_{j+1}^{\mathrm{a}} = f(x_j^{\mathrm{a}}; \theta),$$

and then use this *autoregressively* to predict the future. That is,

$$x_{j+2}^{\mathrm{a}} = f(f(x_j^{\mathrm{a}}; \theta); \theta),$$

etc.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.
- Lower error compared to observations alone.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.
- Lower error compared to observations alone.
- Provided gridded in space and at regular time intervals. This is simpler for ML to deal with.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.
- Lower error compared to observations alone.
- Provided gridded in space and at regular time intervals. This is simpler for ML to deal with.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.
- Lower error compared to observations alone.
- Provided gridded in space and at regular time intervals. This is simpler for ML to deal with.

Why use reanalysis rather than just model forecasts?

Training on model forecasts can only be as good as the physics-based model. May be faster though.

Learning model from reanalysis

Why use reanalysis rather than just observations?

- The reanalysis functions as a physics-based interpolator, fills in information where observations missing.
- Lower error compared to observations alone.
- Provided gridded in space and at regular time intervals. This is simpler for ML to deal with.

Why use reanalysis rather than just model forecasts?

Training on model forecasts can only be as good as the physics-based model. May be faster though.

Downsides:

The reanalysis still includes some error due to the model used to generate it, so the ML model is learning this.

Learning model from reanalysis

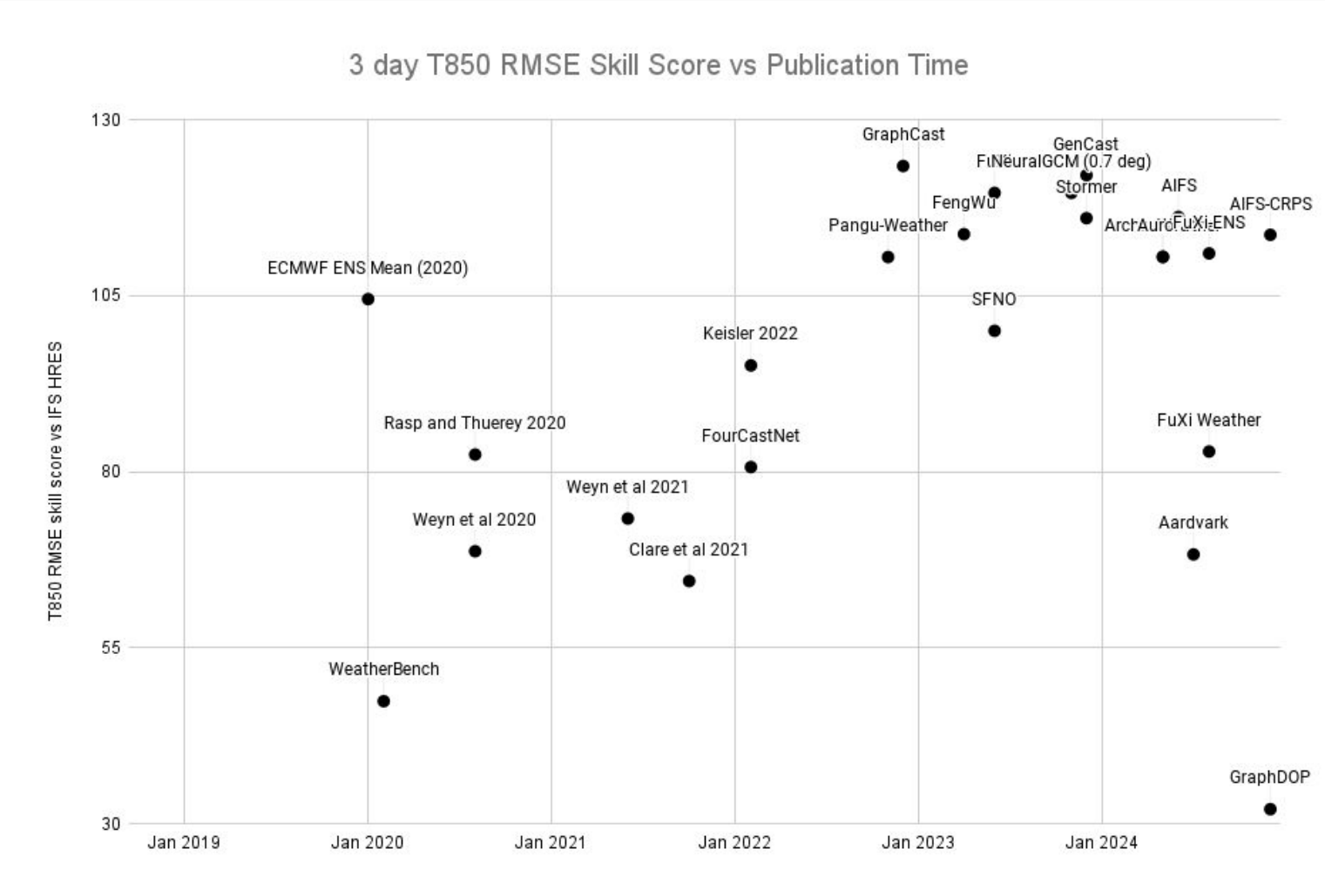

Figure 1: By Stephan Rasp

Learning corrections from analysis increments

Kalman-based DA algorithms can be written as

$$x^{\mathrm{a}} = x^{\mathrm{f}} + K(y - h(x^{\mathrm{f}})).$$

Learning corrections from analysis increments

Kalman-based DA algorithms can be written as

$$x^{\mathrm{a}} = x^{\mathrm{f}} + K(y - h(x^{\mathrm{f}})).$$

Observations are usually assumed to be unbiased ($\mathbb{E}[y - h(x^{\mathrm{t}})] = 0$). Thus, the bias in the analysis is usually smaller than that of the forecast.

Learning corrections from analysis increments

Kalman-based DA algorithms can be written as

$$x^{\mathrm{a}} = x^{\mathrm{f}} + K(y - h(x^{\mathrm{f}})).$$

Observations are usually assumed to be unbiased ($\mathbb{E}[y - h(x^{\mathrm{t}})] = 0$). Thus, the bias in the analysis is usually smaller than that of the forecast.

Thus, the *analysis increment*, $x^{\mathrm{a}} - x^{\mathrm{f}}$, tends to correct for bias.

Learning corrections from analysis increments

Kalman-based DA algorithms can be written as

$$x^{\mathrm{a}} = x^{\mathrm{f}} + K(y - h(x^{\mathrm{f}})).$$

Observations are usually assumed to be unbiased ($\mathbb{E}[y - h(x^{\mathrm{t}})] = 0$). Thus, the bias in the analysis is usually smaller than that of the forecast.

Thus, the *analysis increment*, $x^{\mathrm{a}} - x^{\mathrm{f}}$, tends to correct for bias.

Given a history of analysis increments,

$$\{x_j^{\mathrm{a}} - x_j^{\mathrm{f}}\}_{j=1}^{J},$$

can we extract systematic patterns?

Learning corrections from analysis increments

Kalman-based DA algorithms can be written as

$$x^{\mathrm{a}} = x^{\mathrm{f}} + K(y - h(x^{\mathrm{f}})).$$

Observations are usually assumed to be unbiased ($\mathbb{E}[y - h(x^{\mathrm{t}})] = 0$). Thus, the bias in the analysis is usually smaller than that of the forecast.

Thus, the *analysis increment*, $x^{\mathrm{a}} - x^{\mathrm{f}}$, tends to correct for bias.

Given a history of analysis increments,

$$\{x_j^{\mathrm{a}} - x_j^{\mathrm{f}}\}_{j=1}^{J},$$

can we extract systematic patterns?

If so, we would like to be able to predict this correction before gathering observations.

Learning corrections from analysis increments

We can train an ML model to predict the analysis increment, by regression on the pairs

$$\{(x_j^{\mathrm{f}}, x_j^{\mathrm{a}} - x_j^{\mathrm{f}})\}_{j=1}^{J}.$$

Learning corrections from analysis increments

We can train an ML model to predict the analysis increment, by regression on the pairs

$$\{(x_j^{\mathrm{f}}, x_j^{\mathrm{a}} - x_j^{\mathrm{f}})\}_{j=1}^{J}.$$

Call the predicted analysis increment $\widetilde{\mathrm{AI}}$. Then, when assimilating, we update the background,

$$x_j^{\mathrm{a}} = (x_j^{\mathrm{f}} + \widetilde{\mathrm{AI}}_j) + K_j(y_j - h(x_j^{\mathrm{f}})).$$

Learning forecast model

Methods:

- General: expectation maximization Brajard et al. 2020; Bocquet et al. 2020

Learning forecast model

Methods:

- General: expectation maximization Brajard et al. 2020; Bocquet et al. 2020
- General: auto-differentiable EnKF (Chen, Sanz-Alonso, and Willett 2021)

Learning forecast model

Methods:

- General: expectation maximization Brajard et al. 2020; Bocquet et al. 2020
- General: auto-differentiable EnKF (Chen, Sanz-Alonso, and Willett 2021)
- General: augmented EnKF (Gottwald and Reich 2021)

Learning forecast model

Methods:

- General: expectation maximization Brajard et al. 2020; Bocquet et al. 2020
- General: auto-differentiable EnKF (Chen, Sanz-Alonso, and Willett 2021)
- General: augmented EnKF (Gottwald and Reich 2021)
- Model error correction: analysis increments (Danforth, Kalnay, and Miyoshi 2007; Farchi et al. 2021)

Learning forecast model

Methods:

- General: expectation maximization Brajard et al. 2020; Bocquet et al. 2020
- General: auto-differentiable EnKF (Chen, Sanz-Alonso, and Willett 2021)
- General: augmented EnKF (Gottwald and Reich 2021)
- Model error correction: analysis increments (Danforth, Kalnay, and Miyoshi 2007; Farchi et al. 2021)
- Model error covariance estimation: innovation-based methods (Tandeo et al. 2020)

Using learned forecast model

Using learned forecast model

Assume we have access to an approximate forecast model learned using ML, $\mathcal{M}^a$, which is much faster to evaluate.

Using learned forecast model

Assume we have access to an approximate forecast model learned using ML, $\mathcal{M}^a$, which is much faster to evaluate.

We can then obtain an approximate forecast using $\mathcal{M}^a$.

Using learned forecast model

Assume we have access to an approximate forecast model learned using ML, $\mathcal{M}^a$, which is much faster to evaluate.

We can then obtain an approximate forecast using $\mathcal{M}^a$.

We can state theorems that say that if the models are “close” (in a precise way), the analysis will also be close.

Using learned forecast model

Suppose that we have access to both $\mathcal{M}$ and $\mathcal{M}^a$. $\mathcal{M}$ is a costly "high-fidelity" model which we will take to be the true one, while $\mathcal{M}^a$ is a cheaper "low-fidelity" model.

Using learned forecast model

Suppose that we have access to both $\mathcal{M}$ and $\mathcal{M}^a$. $\mathcal{M}$ is a costly "high-fidelity" model which we will take to be the true one, while $\mathcal{M}^a$ is a cheaper "low-fidelity" model.

We would like to use a large ensemble of $\mathcal{M}^a$ to augment a smaller ensemble of $\mathcal{M}$ and improve ensemble filtering.

Using learned forecast model

Suppose that we have access to both $\mathcal{M}$ and $\mathcal{M}^a$. $\mathcal{M}$ is a costly "high-fidelity" model which we will take to be the true one, while $\mathcal{M}^a$ is a cheaper "low-fidelity" model.

We would like to use a large ensemble of $\mathcal{M}^a$ to augment a smaller ensemble of $\mathcal{M}$ and improve ensemble filtering.

The basic idea of multifidelity Monte Carlo methods is to use a correlated random variable, which may have a different expectation, to reduce the variance in a Monte Carlo estimate.

Using learned forecast model

Suppose that we have access to both $\mathcal{M}$ and $\mathcal{M}^a$. $\mathcal{M}$ is a costly "high-fidelity" model which we will take to be the true one, while $\mathcal{M}^a$ is a cheaper "low-fidelity" model.

We would like to use a large ensemble of $\mathcal{M}^a$ to augment a smaller ensemble of $\mathcal{M}$ and improve ensemble filtering.

The basic idea of multifidelity Monte Carlo methods is to use a correlated random variable, which may have a different expectation, to reduce the variance in a Monte Carlo estimate.

Alternate approach: multi-model ensemble Kalman filter (Bach and Ghil 2023)

Using learned forecast model

Suppose that we have K low-fidelity models $\{\mathcal{M}_k\}_{k=1}^K$. $\mathcal{M}_{\text{hi}}$ is the high-fidelity model, with respect to which we would like to be unbiased.

Using learned forecast model

Suppose that we have K low-fidelity models $\{\mathcal{M}_k\}_{k=1}^K$. $\mathcal{M}_{\text{hi}}$ is the high-fidelity model, with respect to which we would like to be unbiased.

Assume furthermore that the ensemble sizes for each model are arranged such that $0 < N_{\text{hi}} \leq N_1 \leq \ldots \leq N_K$.

Using learned forecast model

Suppose that we have K low-fidelity models $\{\mathcal{M}_k\}_{k=1}^K$. $\mathcal{M}_{\text{hi}}$ is the high-fidelity model, with respect to which we would like to be unbiased.

Assume furthermore that the ensemble sizes for each model are arranged such that $0 < N_{\text{hi}} \leq N_1 \leq \ldots \leq N_K$.

Draw random samples $\{v^{(n)}\}_{n=1}^{N_K}$. Then compute the following ensemble means:

$$\hat{m}_{\text{hi}} = \frac{1}{N_{\text{hi}}} \sum_{n=1}^{N_{\text{hi}}} \mathcal{M}_{\text{hi}}(v^{(n)}), \qquad \hat{m}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} \mathcal{M}_k(v^{(n)}),$$

$$\hat{m}'_k = \frac{1}{N_{k-1}} \sum_{n=1}^{N_{k-1}} \mathcal{M}_k(v^{(n)}).$$

Using learned forecast model

Then we consider a multifidelity estimator given by

$$\hat{m} = \hat{m}_{\text{hi}} + \sum_{k=1}^{K} \alpha_k (\hat{m}_k - \hat{m}'_k), \quad \alpha_k = \frac{\rho_k Q_{\text{hi}}}{Q_k}, \tag{10}$$

where ρ_k is the correlation coefficient between $\mathcal{M}_{\text{hi}}$ and $\mathcal{M}_k$, and Q_k^2, Q_{hi}^2 are the variances of $\mathcal{M}_k$ and $\mathcal{M}_{\text{hi}}$, respectively.

Theorem

The estimator is unbiased with respect to $\hat{m}_{\text{hi}}$, and has variance

$$\text{Var}[\hat{m}] = \frac{Q_{\text{hi}}^2}{N_{\text{hi}}} - \sum_{k=1}^{K} \left(N_{k-1}^{-1} - N_k^{-1} \right) \rho_k^2 Q_{\text{hi}}^2. \tag{11}$$

If $\rho_k^2 > 0$ for any k, $\text{Var}[\hat{m}] < \text{Var}[\hat{m}_{\text{hi}}]$.

Using learned forecast model

We can use the multifidelity Monte Carlo estimator in the forecast step of, e.g., an ensemble Kalman filter.

Using learned forecast model

We can use the multifidelity Monte Carlo estimator in the forecast step of, e.g., an ensemble Kalman filter.

Given a fixed computational budget, the ensemble sizes for the different models can be optimized to give the lowest variance.

Learning filters/state estimators

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Why? Can do this to get a trajectory closer to the truth.

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Why? Can do this to get a trajectory closer to the truth.

Can also learn an existing DA algorithm for computational efficiency using supervised learning.

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Why? Can do this to get a trajectory closer to the truth.

Can also learn an existing DA algorithm for computational efficiency using supervised learning.

Example 1: learning inflation and localization in an EnKF.

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Why? Can do this to get a trajectory closer to the truth.

Can also learn an existing DA algorithm for computational efficiency using supervised learning.

Example 1: learning inflation and localization in an EnKF.

Example 2: learning a gain matrix in $x_j = \hat{v}_j + K(y_j^\dagger - h(\hat{v}_j))$.

Learning filters/state estimators

Suppose now that we want to learn the analysis step, or certain unknown parts of the analysis step.

Why? Can do this to get a trajectory closer to the truth.

Can also learn an existing DA algorithm for computational efficiency using supervised learning.

Example 1: learning inflation and localization in an EnKF.

Example 2: learning a gain matrix in $x_j = \hat{v}_j + K(y_j^\dagger - h(\hat{v}_j))$.

Example 3: learning an analysis step for an ensemble filter, parameterized as a neural network.

Learning state estimators

Instead of trying to match the filtering distribution, we can instead try to match the true trajectory.

Learning state estimators

Instead of trying to match the filtering distribution, we can instead try to match the true trajectory.

Assuming we have access to the model, we can simulate a trajectory x^{t} and observations $Y^{\dagger}$, and $x_j(Y_j^{\dagger}; \theta)$ to be some estimate of x_j^{t}. Then,

$$\mathrm{J}(\theta) = \frac{1}{J} \sum_{j=1}^{J} \|x_j(Y_j^{\dagger}; \theta) - x_j^{\mathrm{t}}\|^2, \tag{12a}$$

$$\theta^* = \arg\min_{\theta} \mathrm{J}(\theta). \tag{12b}$$

Learning state estimators

Suppose now that we do not have access to the true model, and hence have to minimize a loss against observations $Y^\dagger$:

$$\mathrm{J}(\theta) = \frac{1}{J} \sum_{j=1}^{J} \| h(x_j(Y_j^\dagger; \theta)) - y_j^\dagger \|^2, \tag{13a}$$

$$\theta^* = \arg\min_\theta \mathrm{J}(\theta). \tag{13b}$$

Learning state estimators

Suppose now that we do not have access to the true model, and hence have to minimize a loss against observations $Y^\dagger$:

$$\mathrm{J}(\theta) = \frac{1}{J}\sum_{j=1}^{J} \|h(x_j(Y_j^\dagger;\theta)) - y_j^\dagger\|^2, \tag{13a}$$

$$\theta^* = \arg\min_\theta \mathrm{J}(\theta). \tag{13b}$$

Since x_j is a function of $y_j^\dagger$, this cost function can cause x_j to be overfit to observations.

Learning state estimators

Suppose now that we do not have access to the true model, and hence have to minimize a loss against observations $Y^\dagger$:

$$\mathrm{J}(\theta) = \frac{1}{J}\sum_{j=1}^{J} \|h(x_j(Y_j^\dagger;\theta)) - y_j^\dagger\|^2, \tag{13a}$$

$$\theta^* = \arg\min_\theta \mathrm{J}(\theta). \tag{13b}$$

Since x_j is a function of $y_j^\dagger$, this cost function can cause x_j to be overfit to observations.

Example: if h is surjective, one can achieve a perfect score by simply setting $x_j = h^{-1}(y_j^\dagger)$, where h^{-1} is the right inverse of h.

Learning state estimators

Taking the expectation over the observation noise realization, $\eta_j^\dagger \sim \mathcal{N}(0, R)$, we obtain (Mallia-Parfitt and Bröcker 2016):

$$\mathbb{E}[\|h(x_j(\theta)) - y_j^\dagger\|^2] = \mathbb{E}[\|h(x_j) - h(x_j^\dagger)\|^2] + \mathrm{tr}(R) - 2\mathbb{E}[h(x_j(\theta))^\top \eta_j^\dagger].$$

Learning state estimators

Taking the expectation over the observation noise realization, $\eta_j^\dagger \sim \mathcal{N}(0, R)$, we obtain (Mallia-Parfitt and Bröcker 2016):

$$\mathbb{E}[\|h(x_j(\theta)) - y_j^\dagger\|^2] = \mathbb{E}[\|h(x_j) - h(x_j^\dagger)\|^2] + \mathrm{tr}(R) - 2\mathbb{E}[h(x_j(\theta))^\top \eta_j^\dagger].$$

$\mathbb{E}[h(x_j)^\top \eta_j^\dagger]$ will generally be positive, since the observation $y_j^\dagger$ was used to produce the analysis x_j. The first term on the RHS is what we would like to estimate.

Learning state estimators

Taking the expectation over the observation noise realization, $\eta_j^\dagger \sim \mathcal{N}(0, R)$, we obtain (Mallia-Parfitt and Bröcker 2016):

$$\mathbb{E}[\|h(x_j(\theta)) - y_j^\dagger\|^2] = \mathbb{E}[\|h(x_j) - h(x_j^\dagger)\|^2] + \mathrm{tr}(R) - 2\mathbb{E}[h(x_j(\theta))^\top \eta_j^\dagger].$$

$\mathbb{E}[h(x_j)^\top \eta_j^\dagger]$ will generally be positive, since the observation $y_j^\dagger$ was used to produce the analysis x_j. The first term on the RHS is what we would like to estimate.

$\mathbb{E}[\|h(x_j(\theta)) - y_j^\dagger\|^2] + 2\mathbb{E}[h(x_j(\theta))^\top \eta_j^\dagger]$ is thus a better proxy for out-of-sample performance, and should be minimized instead of only $\mathbb{E}[\|h(x_j(\theta)) - y_j^\dagger\|^2]$.

References i

- Bach, E. and M. Ghil (2023). **“A Multi-Model Ensemble Kalman Filter for Data Assimilation and Forecasting”.** *Journal of Advances in Modeling Earth Systems.*
- Bocquet, M., J. Brajard, A. Carrassi, and L. Bertino (2020). **“Bayesian Inference of Chaotic Dynamics by Merging Data Assimilation, Machine Learning and Expectation-Maximization”.** *Foundations of Data Science.*
- Boudier, P., A. Fillion, S. Gratton, S. Gürol, and S. Zhang (2023). **“Data Assimilation Networks”.** *Journal of Advances in Modeling Earth Systems.*
- Brajard, J., A. Carrassi, M. Bocquet, and L. Bertino (2020). **“Combining Data Assimilation and Machine Learning to Emulate a Dynamical Model from Sparse and Noisy Observations: A Case Study with the Lorenz 96 Model”.** *Journal of Computational Science.*

References ii

- Chen, Y., D. Sanz-Alonso, and R. Willett (2021). **"Auto-Differentiable Ensemble Kalman Filters".** arXiv: 2107.07687 [cs, stat].
- Danforth, C. M., E. Kalnay, and T. Miyoshi (2007). **"Estimating and Correcting Global Weather Model Error".** *Monthly Weather Review*.
- Farchi, A., P. Laloyaux, M. Bonavita, and M. Bocquet (2021). **"Using Machine Learning to Correct Model Error in Data Assimilation and Forecast Applications".** *Quarterly Journal of the Royal Meteorological Society*.
- Gottwald, G. A. and S. Reich (2021). **"Supervised Learning from Noisy Observations: Combining Machine-Learning Techniques with Data Assimilation".** *Physica D: Nonlinear Phenomena*.

References iii

- Hammoud, M. A. E. R., N. Raboudi, E. S. Titi, O. Knio, and I. Hoteit (2024). ***Data Assimilation in Chaotic Systems Using Deep Reinforcement Learning***. arXiv: `2401.00916 [physics]`. Pre-published.
- Mallia-Parfitt, N. and J. Bröcker (2016). **"Assessing the Performance of Data Assimilation Algorithms Which Employ Linear Error Feedback"**. *Chaos: An Interdisciplinary Journal of Nonlinear Science*.
- Rozet, F. and G. Louppe (2023). **"Score-Based Data Assimilation"**. *Advances in Neural Information Processing Systems*.
- Spantini, A., R. Baptista, and Y. Marzouk (2022). **"Coupling Techniques for Nonlinear Ensemble Filtering"**. *SIAM Review*.

References iv

- Tandeo, P., P. Ailliot, M. Bocquet, A. Carrassi, T. Miyoshi, M. Pulido, and Y. Zhen (2020). **“A Review of Innovation-Based Methods to Jointly Estimate Model and Observation Error Covariance Matrices in Ensemble Data Assimilation”.** *Monthly Weather Review.*
- Waller, J. A., S. L. Dance, and N. K. Nichols (2016). **“Theoretical Insight into Diagnosing Observation Error Correlations Using Observation-Minus-Background and Observation-Minus-Analysis Statistics”.** *Quarterly Journal of the Royal Meteorological Society.*