Completing physics-based models by learning hidden dynamics through data assimilation

Arthur Filoche, Julien Brajard, Anastase Alexandre Charantonis, Dominique Béréziat

Completing physics-based model by learning hidden dynamics through data assimilation

Arthur Filoche
Sorbonne Université - LIP6, Paris
arthur.filoche@lip6.fr

Julien Brajard
Sorbonne Université - LOCEAN, Paris
NERSC, Bergen

Anastase Charantonis
Sorbonne Université - LOCEAN, Paris
ENSIEE, Évry

Dominique Béréziat
Sorbonne Université - LIP6, Paris

Abstract

Data Assimilation remains the operational choice when it comes to forecast and estimate Earth's dynamical systems. The analogy with Machine Learning has already been shown and is still being investigated to address the problem of improving physics-based models. Even though both techniques learn from data, machine learning focuses on inferring models while data assimilation concentrates on hidden system state estimation with the help of a dynamical model. In this work, we exploit the complementarity of these methods in a twin experiment where the system is partially observed and the known dynamics are incomplete. Finally, we succeed in partially retrieving the dynamics of a fully-unobserved variable by training a hybrid model through variational data assimilation.

1 Introduction

Geosciences have a long standing experience in modeling, forecasting or estimating complex dynamical systems like atmospheric or ocean dynamics which present high dimensionalities and non-linearities. Most of these models came from physical laws and are described as partial differential equations ultimately used in their discretized version. Even with hypothetically known initial conditions, imperfection and chaoticity of such models enforce the use of observations over time. Data Assimilation (DA) [1] proposes a large panel of methods to optimally combine a dynamical model and observations allowing to predict, filter, or smooth system state trajectory. Those methods are the ones operationally used in numerical weather prediction [2].

Even though physical modeling has proven to be extremely useful, recent advances in Machine Learning (ML) deserve consideration. Neural networks and more precisely ResNet-like architectures can be seen as dynamical systems and numerical schemes, respectively [3]. They are now considered state of the art in a vast amount of tasks involving spatio-temporal forecasting which makes them very appealing for numerical weather prediction [4, 5]. But to train such networks, one needs dense and representative data which is rarely the case in Earth sciences. At the same time, DA offers a proper framework allowing to learn from partial, noisy, and indirect observations. Thus, each of this field can profit from the other by providing either a learnable class of dynamical models or dense data sets.

Approaches combining DA and ML have naturally emerged [6] and already shown promising results on small-sized system like those of Lorenz [7]. It was also argued that, under certain conditions, alternating DA and ML steps can be seen as an Expectation-Maximization algorithm [8, 9, 10]

In this work we propose to use such methods on a relatively high-dimensional system where the physics is partially known. We aim at completing this base model by learning the dynamics of a

AI for Earth Sciences Workshop at NeurIPS 2020.

fully-unobserved variable through data assimilation. Also, we benefit from powerful and flexible tools provided by the deep learning community based on automatic differentiation that are clearly suitable for variational data assimilation [11], avoiding explicit adjoint modeling.

2 Framework

2.1 The control problem

A system state $\mathbf{X}$ evolves over discrete time t according to a parameterized dynamical Markov model $\mathbb{M}_\theta$. Partial and noisy observations $\mathbf{Y}$ are available through an observation operator $\mathbb{H}$ as represented in Figure 1. The system state trajectory $[\mathbf{X}_0, .., \mathbf{X}_T]$ and the model parameters $\boldsymbol{\theta}$ are the quantities to be estimated.

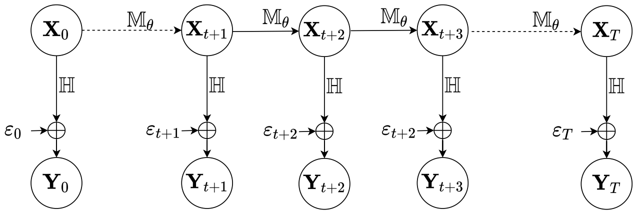

Figure 1: System representation as a parametric hidden Markov model.

In a variational formalism [12] this can be expressed through the minimization of an energy function of the general form $J(\mathbf{X}, \boldsymbol{\theta}) = \sum_t \|\mathbf{Y}_t - \mathbb{H}\mathbf{X}_t\| + \sum_t \|\mathbf{X}_{t+1} - \mathbb{M}_\theta(\mathbf{X}_t)\|$, where the first term is the observation error and the second the model error. One major difficulty of such optimization problem arises from the fact that the control variables are of different natures. Usually, system states are estimated over a temporal window which may not include enough examples to train a machine learning model and more particularly a neural network. We may consider several trajectories $\left[\mathbf{X}_0^i, .., \mathbf{X}_T^i\right]$ even though $\boldsymbol{\theta}$ should not depend on a particular one. Ultimately, we choose to avoid the excessively ill-posed joint optimization problem.

2.2 Coordinate descent: Alternate DA and ML

When $\boldsymbol{\theta}$ is known, DA methods combining $\mathbb{M}_\theta$ and $\mathbf{Y}$ can produce state estimation $\hat{\mathbf{X}}$. On the flip side, when the system is fully observed with total confidence, $\boldsymbol{\theta}$ can be learned by regression. Alternating DA and ML steps, we successively optimize along $\mathbf{X}$ and $\boldsymbol{\theta}$. This is why we refer to the algorithm we use as *coordinate descent* (see Algorithm 1).

Algorithm 1 Coordinate descent

```
Initialize $\boldsymbol{\theta}$
while Convergence do
    for every trajectory $i$ do
        States estimation $\hat{\mathbf{X}}^i$ minimizing $\sum_t \|\mathbf{Y}_t^i - \mathbb{H}\mathbf{X}_t^i\|$
    Learning $\boldsymbol{\theta}$ minimizing $\sum_{i,t} \|\mathbf{X}_{t+1}^i - \mathbb{M}_\theta(\mathbf{X}_t^i)\|$
```

3 Case Study

3.1 Materials

The system state evolves over a discretized space-time domain $\Omega \times [0 : T]$ where Ω is a bounded rectangle of $\mathbb{R}^2$. At each time $\mathbf{X}_t = [I_t, \mathbf{w}_t] \in \mathbb{R}^{3\times 64\times 64}$ is composed of a passive tracer I_t associated to a motion field $\mathbf{w}_t$.

Ground truth

Synthetic data are generated by integrating an advection model over several initial conditions and constitute a ground truth data-set of multiple system trajectories. I_0 is always the same 2D-sine wave whose statistics are similar to those of sea surface temperatures, whereas for initial motion fields $\mathbf{w}_0^i$, we use the ocean surface circulation in different areas of the North Atlantic. The motion field transports the passive tracer and also itself. An example of a simulated trajectory is shown in Fig. 2.

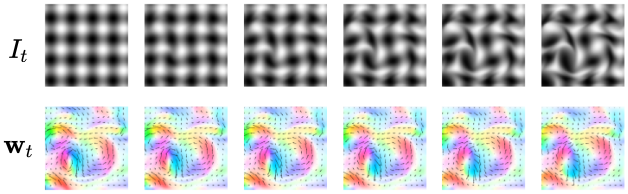

Figure 2: System trajectory simulated with the ground truth dynamics.

Available information

In a realistic Earth science setup, available information is limited: physics-based models are incomplete and observations are partial and noisy. In our specific case only the passive tracer I is observed with an additive noise: $\mathbf{Y}_t = I_t = \mathbb{H}\mathbf{X}_t + \varepsilon_{R_t}$ where ε_{R_t} is a Gaussian white noise of known covariance matrix $\mathbf{R}_t$. We note $d_M^2(x,y) = \|x-y\|_{R_t}^2 = \langle (x-y)^\top | \mathbf{R}_t^{-1}(x-y)\rangle$ the associated Mahalanobis distance. Regarding our knowledge about the underlying dynamics, we assume to know the full model except for evolution equation on the motion field.

Introducing our hybrid model

Our main goal is therefore to recover this missing dynamics of the motion field which is never observed. To do so we introduce in Eq. 1, a parameterized hybrid dynamics $\mathbb{M}_\theta$ that combines a numerical scheme [13] representing the known physics $\mathbb{M}_P$ with a fully convolutional neural network $\mathbb{M}_L(\boldsymbol{\theta})$ to be trained to represent the unresolved part of the ground truth dynamics. In the perfect case, $\boldsymbol{\theta}^*$ is such that $f_{\theta^*}(\mathbf{w}) = \mathbf{w}.\nabla\mathbf{w}$.

$$\mathbb{M}_P + \mathbb{M}_L(\boldsymbol{\theta}) = \mathbb{M}_\theta \quad \text{the resolvent of the following PDE-system} \quad \begin{cases} \dfrac{\partial I}{\partial t} + \mathbf{w}.\nabla I = 0 \\ \dfrac{\partial \mathbf{w}}{\partial t} + f_\theta(\mathbf{w}) = 0 \\ \nabla I = 0, \quad \partial\Omega \\ \mathbf{w} = 0, \quad \partial\Omega \end{cases} \tag{1}$$

3.2 Learning scheme

As depicted before and schematized in Figure 3, we will use a coordinate descent approach alternating assimilation and learning steps in order to ultimately train the hybrid model. To build a consequent enough training set, several trajectories are assimilated. During assimilation steps $\boldsymbol{\theta}$ is fixed while during learning steps $\mathbf{X}$ is fixed. We initialize the procedure with the incomplete physics-based model which means that the first assimilation step is equivalent to a well known variational optical flow estimation [14]. The next assimilation steps are performed with an evolution model of the motion field as in [15].

Data Assimilation step: strong-constraint 4D-Var

States estimation is achieved on a sliding time window of size w with the 4D-Var algorithm which aims at solving the PDE-constrained optimization problem described in Eq. 2. While weak-constraint 4D-Var, allowing model errors, generally produces better results, we intentionally chose to use the strong version, assuming a perfect model, as we represent this model error with a parameterized dynamics. Regularization parameters α and β are tuned using sequential model-based optimization [16] over forecast performance after assimilation.

$$
\begin{aligned}
\min_{\mathbf{X}_t} \quad & J_{DA}(\mathbf{X}_t;\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=t}^{t+w-1} \|\mathbf{Y}_i - \mathbb{H}\mathbf{X}_i\|^2_{R_i} + \frac{\alpha}{2}\|\nabla \mathbf{w}_t\|^2_2 + \frac{\beta}{2}\|\nabla . \mathbf{w}_t\|^2_2 \\
\text{s.t.} \quad & \mathbf{X}_{i+1} = \mathbb{M}_\theta(\mathbf{X}_i)
\end{aligned}
\tag{2}
$$

Machine Learning step

The ML step is a regression on estimated states minimizing $J_{ML}(\hat{\mathbf{X}}_t, \hat{\mathbf{X}}_{t+1}) = \|\hat{\mathbf{w}}_{t+1} - (\hat{\mathbf{w}}_t + \int_t^{t+1} f_\theta(\hat{\mathbf{w}}_t))\|^2_2$. The neural network is trained by stochastic gradient descent and employs batch normalization, 3×3 kernel convolution and ReLu activation.

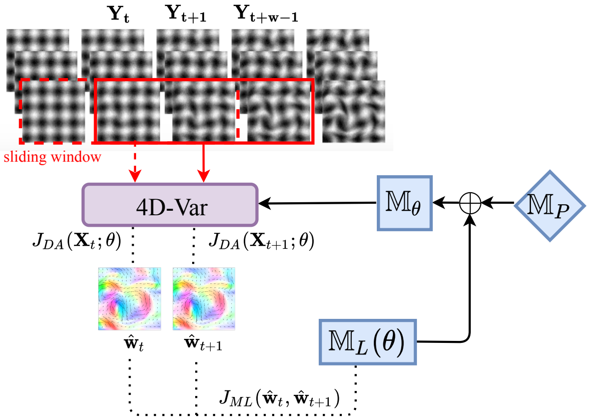

Figure 3: Schematic view of the learning scheme.

4 Results – forecast skill

To evaluate the trained hybrid model $\mathbb{M}_\theta$, we produce forecasts over multiple initial conditions which were not used during the training and compare them with ground truth trajectories by calculating RMSE on the tracer I. We benchmark the obtained hybrid model against the incomplete physics-based model $\mathbb{M}_P$ and a hybrid model trained on perfect motion field data $\mathbb{M}^p_\theta$ from ground truth simulation usually unknown. In Fig. 4, we see that $\mathbb{M}_\theta$ outperforms the physics-based model $\mathbb{M}_P$ and is relatively close to $\mathbb{M}^p_\theta$ whose performance is only limited by the network architecture choice. Also, reduced uncertainty indicates ability to generalize particularly in areas with intense motion.

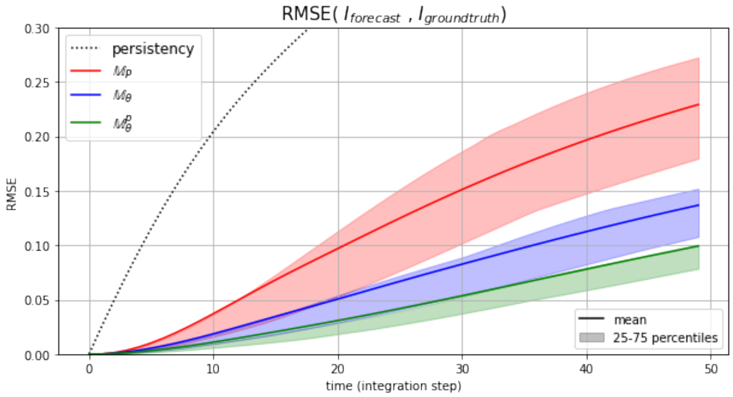

Figure 4: Comparison between $\mathbb{M}_P$, $\mathbb{M}_\theta$ and $\mathbb{M}^p_\theta$ against the ground truth dynamics.

5 Conclusion

We introduced a hybrid model with learnable components. To train it, we leveraged DA ability to learn from sparse and noisy observations with the help of deep learning tools based on automatic differentiation. Finally we tested it in a twin experiment where we succeeded in partially retrieving a missing dynamics of a fully-unobserved variable in a relatively high-dimensional system.

References

[1] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018.

[2] Jennifer Waters, Daniel J. Lea, Matthew J. Martin, Isabelle Mirouze, Anthony Weaver, and James While. Implementing a variational data assimilation system in an operational 1/4 degree global ocean model. *Quarterly Journal of the Royal Meteorological Society*, 141(687):333–349, January 2015.

[3] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017.

[4] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat. Deep learning and process understanding for data-driven earth system science. *Nat.*, 566(7743):195–204, 2019.

[5] Stephan Rasp, Peter D. Dueben, Sebastian Scher, Jonathan A. Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: A benchmark dataset for data-driven weather forecasting, 2020.

[6] Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino. Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear Processes in Geophysics*, 26(3):143–162, 2019.

[7] 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. *Geoscientific Model Development Discussions*, 2019:1–21, 2019.

[8] Zoubin Ghahramani and Sam Rowis. Learning nonlinear dynamical systems using an em algorithm. *Adv. Neural Inf. Processing Syst.*, 11(1), 03 1999.

[9] Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *Foundations of Data Science*, 2(1):55–80, 2020.

[10] Duong Nguyen, Said Ouala, Lucas Drumetz, and Ronan Fablet. Assimilation-based Learning of Chaotic Dynamical Systems from Noisy and Partial Data. In *ICASSP 2020 : International Conference on Acoustics, Speech, and Signal Processing*, 2020.

[11] Olivier Talagrand François-Xavier Le Dimet. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus*, 38A(10):97, 1986.

[12] Y. Trémolet. Accounting for an imperfect model in 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 132:2483–2504, 2006.

[13] Andrew Staniforth and Jean Côté. Semi-Lagrangian Integration Schemes for Atmospheric Models—A Review. *Monthly Weather Review*, 119(9):2206–2223, 09 1990.

[14] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[15] Dominique Béréziat and Isabelle Herlin. Motion and Acceleration from Image Assimilation with evolution models. *Digital Signal Processing*, 83:45–58, 2018.

[16] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523, 2011.

[17] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[18] Yann Lepoittevin and Isabelle Herlin. Regularization terms for motion estimation. Links with spatial correlations. In *VISAPP - International Conference on Computer Vision Theory and Applications*, volume 3, pages 458–466, 2016.

Acknowledgement

CO2 Emission Related to Experiments:

Experiments were conducted using a private infrastructure, which has a carbon efficiency of $0.053\,\mathrm{kg}$ CO$_2$eq $/(\mathrm{kW\,h})$. A cumulative of 24 hours of computation was performed on hardware of type Tesla P100 (TDP of 250W).

Total emissions are estimated to be $0.32\,\mathrm{kg}$ CO$_2$eq.

Estimations were conducted using the MachineLearning Impact calculator presented in [17].

A Experiment details

A.1 Ground truth

Initial conditions

Initial motion fields came from an ocean circulation model re-analysis where they are expressed in $\mathrm{m\,s^{-1}}$. Different used zones exposed in Figure. 5 constitute the diversity of the dataset. Half of the zones are used in the training scheme, the other half is used for testing.

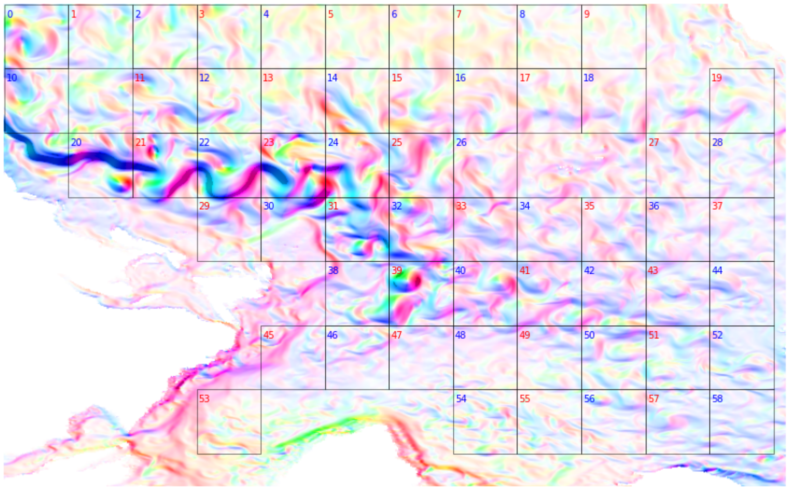

Figure 5: Initial motion fields by zones.

Dynamics

The ground dynamics is an advection model described in the Eq. 3.

$$\begin{cases} \dfrac{\partial I}{\partial t} + \mathbf{w}.\nabla I = 0 \\ \dfrac{\partial \mathbf{w}}{\partial t} + (\mathbf{w}.\nabla)\mathbf{w} = 0 \\ \nabla I = 0, \quad \partial\Omega \\ \mathbf{w} = 0, \quad \partial\Omega \end{cases} \tag{3}$$

Respectively, these equations represent: linear advection of the I tracer by the velocity field, non-linear advection of the velocity by itself, vanishing Neumann boundary conditions for I and vanishing Dirichlet boundary conditions for $\mathbf{w}$. The same semi-Lagrangian numerical scheme [13] is used for both advection equations.

The space-time domain is discretized over a grid of parameter $dx = dy = 10000$ and $dt = 8640$ which means, in relation with measurement unit of the velocity field, that each pixel covers an area of $10\,\mathrm{km}^2$ and each time step represents $8640\,\mathrm{s} = 0.1$ day.

A.2 Data Assimilation: 4D-Var

Usually, PDE-constrained optimization in inverse problems is solved with an adjoint method. Gradients are obtained by retropropagation of the adjoint dynamics which can be problematic to handle. Thanks to deep learning tools based on automatic differentiation and using fully-differentiable dynamical models, we can obtain those gradients with no effort. The used 4D-Var algorithm is detailed in Algorithm 2.

Algorithm 2 Strong 4D-Var

```
Set the iteration index $k = 0$
Initialize the state vector $\mathbf{X}_t^k = \mathbf{X}_{init}$
**forward integration** : run the model $\mathbb{M}_\theta$ and compute $J_{DA}^k$
**backward**: run automatic differentiation and compute $\nabla J_k$
**while** $J_{DA}^k < \epsilon$ and $k < MaxIter$ **do**
    update state vector $\mathbf{X}_t^k$ : $\mathbf{X}_t^{k+1} = BFGS(\mathbf{X}_t^k, J_{DA}^k, \nabla J_{DA}^k)$
    **forward integration** : run the model $\mathbb{M}$ and compute $J_{DA}^{k+1}$
    **backward** : automatic differentiation computes $\nabla J_{DA}^{k+1}$
    set $k = k + 1$
    **return** $\mathbf{X}_t^k$
```

Under Gaussian errors and linear model hypothesis, 4D-Var leads to the maximum a posteriori estimation of the state which motivates the used cost function. Even with the used regularization, enforcing smoothness on the estimated velocity field, the Bayesian interpretation still stands. As depicted in [18], the regularization can be embedded in error covariance matrices as a prior knowledge.

A.3 Machine Learning step

As the function to learn is relatively simple, there was no doubt that classical architecture would be able to approximate it well. We tested several fully convolutional neural networks and best performances were obtained with encoder-decoder based architectures. However, the neural network should be used inside the 4D-Var algorithm which is computationally expensive considering the multiple forward-backward integration required. We made a compromise between the size of the network and its ability to learn the desired function on perfect data and ended up with the following architecture:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 64, 64]             304
            Conv2d-2           [-1, 32, 64, 64]           4,640
       BatchNorm2d-3           [-1, 32, 64, 64]              64
            Conv2d-4           [-1, 16, 64, 64]           4,624
       BatchNorm2d-5           [-1, 16, 64, 64]              32
            Conv2d-6            [-1, 8, 64, 64]           1,160
            Conv2d-7            [-1, 2, 64, 64]              18
================================================================
Total params: 10,842
Trainable params: 10,842
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.03
Forward/backward pass size (MB): 3.81
Params size (MB): 0.04
Estimated Total Size (MB): 3.89
----------------------------------------------------------------
```

The training is performed with a batch size of 32, a decaying learning rate starting at 5×10^{-3}, He normal weights initialization and Adam optimizer.