Data assimilation, machine learning and dynamical systems - Part I

Marc Bocquet[1] **marc.bocquet@enpc.fr** and Julien Brajard[2] **julien.brajard@nersc.no**

(1) CEREA, École des Ponts et EdF R&D, Île-de-France, France

(2) Nansen Center (NERSC), Bergen, Norway

During this session, we will discover some connections between data assimilation and deep learning when applied to dynamical systems.

Synopsis

The goal of this lecture is to give a brief and very limited introduction to the connections between **machine learning/deep learning** and **data assimilation**. Machine learning has found many new convincing applications over the past couple of years, besides computer vision or natural language. The geosciences are among them. Even within the geosciences, there is a considerable range of potential applications of machine learning and deep learning; some of them have been evidenced recently.

Our specific goal today will be to not only learn the state of a physical system through its observation and a prior of this state but also to correct its dynamics.This contrasts with traditional data assimilation where the model is usually assumed to be known, or corrected via only a bunch of parameters in the control variables.

For more information about the ideas developed and exemplified in this mooc, please have a look at the following papers:

- Bocquet, M., Brajard, J., Carrassi, A., & Bertino, L. (2019). Data

assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlin. Processes Geophys.*, *26*, 143--162. **https://doi.org/10.5194/npg-26-143-2019**

- Bocquet, M., Brajard, J., Carrassi, A., & Bertino, L. (2020). Bayesian

inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *Foundations of Data Science*, *2*, 55--80. **https://doi.org/10.3934/fods.2020004**

- Brajard, J., Carrassi, A., Bocquet, M., & Bertino, L. (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. *J. Comput. Sci.*, *44*, 101171. **https://doi.org/10.1016/j.jocs.2020.101171**

- Brajard, J., Carrassi, A., Bocquet, M., & Bertino, L. (2021). Combining

data assimilation and machine learning to infer unresolved scale parametrisation. *Phil. Trans. R. Soc. A*, *379*, 20200086. **https://doi.org/10.1098/rsta.2020.0086**

- Farchi, A., Bocquet, M., Laloyaux, P., Bonavita, M., & Malartic, Q.

(2021). A comparison of combined data assimilation and machine learning methods for offline and online model error correction. *J. Comput. Sci.*, *55*, 101468. **https://doi.org/10.1016/j.jocs.2021.101468**

- Farchi, A., Laloyaux, P., Bonavita, M., & Bocquet, M. (2021). Using

machine learning to correct model error in data assimilation and forecast applications. *Q. J. R. Meteorol. Soc.*, *147*, 3067--3084. **https://doi.org/10.1002/qj.4116**

Preamble: theoretical fundamentals and objective

Dynamical system and observation

Let us consider a dynamical system which is observed:

$$\mathbf{x}_k = M_k(\mathbf{x}_{k-1}) + \eta_k, \quad (1)$$

$$\mathbf{y}_k = H_k(\mathbf{x}_{k-1}) + \varepsilon_k, \quad (2)$$

where $\mathbf{x}_k$ is the state vector at time t_k, $\mathbf{y}_k$ is the observation vector at time t_k, M_k is the numerical evolution model from time t_{k-1} to time t_k, and H_k is the observation operator. The noise vector ε_k follows a Gaussian distribution and is meant to account for some of model error: $\varepsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$. The noise vector η_k follows a Gaussian distribution and is meant to account for observation error: $\eta_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$.

Weak-constraint 4D-Var and generalisation

Then the typical weak-constraint 4D-Var associated to this system is described by the cost function:

$$J(\mathbf{x}_{0:K}) = -J_\mathrm{b}(\mathbf{x}_0) + \sum_{k=0}^{K} \|\mathbf{y}_k - H_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \sum_{k=1}^{K} \|\mathbf{x}_k - M_k\left(\mathbf{x}_{k-1}\right)\|^2_{\mathbf{Q}_k^{-1}},$$

where J_b is the prior/background term, the second term is departure from the observations, and the third term is the model error term which measures the departure from what the model M_k would predict. Now, we can widen the problem and assume the model is at least partly statistical on depend a vector of constant-in-time parameters $\boldsymbol{\omega}$ to be adjusted if not learned, i.e.:

$$\mathbf{x}_k = M_k(\boldsymbol{\omega}, \mathbf{x}_{k-1}) + \eta_k.$$

They could be physical parameters of the model, or weights and biases of a neural network correction to a physical model. The generalised cost function becomes:

$$J(\boldsymbol{\omega}, \mathbf{x}_{0:K}) = -J_\mathrm{b}(\mathbf{x}_0) + \sum_{k=0}^{K} \|\mathbf{y}_k - H_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \sum_{k=1}^{K} \|\mathbf{x}_k - M_k\left(\boldsymbol{\omega}, \mathbf{x}_{k-1}\right)\|^2_{\mathbf{Q}_k^{-1}}.$$

Combined data assimilation and machine learning

First step: data assimilation

Hence, we want to minimise this cost function both on the trajectory $\mathbf{x}_{0:K}$ and on the model statistical parameters. But they are very different in nature and quite difficult to minimise altogether. Hence we will split the task on first a data assimilation part focusing on $\mathbf{x}_k$, and then a machine learning task focusing on $\boldsymbol{\omega}$. We first assime the model is of the form:

$$M_k(\boldsymbol{\omega}, \mathbf{x}_{k-1}) = \Phi(\mathbf{x}_{k-1}) + \mathrm{nn}(\boldsymbol{\omega}, \mathbf{x}_{k-1})$$

where Φ is a known physical model, approximation of the true model and $\mathrm{nn}(\boldsymbol{\omega})$ is a neural network correction.

$$J(\boldsymbol{\omega}, \mathbf{x}_{0:K}) = -J_\mathrm{b}(\mathbf{x}_0) + \sum_{k=0}^{K} \|\mathbf{y}_k - H_k(\mathbf{x}_k)\|^2_{\mathbf{R}_k^{-1}} + \sum_{k=1}^{K} \|\mathbf{x}_k - M_k\left(\boldsymbol{\omega}, \mathbf{x}_{k-1}\right)\|^2_{\mathbf{Q}_k^{-1}}.$$

At fixed $\boldsymbol{\omega}$, this is a smoothing data assimilation problem, which can be solved more or less acurately by any proper data assimilation method: 4D-Var, WC 4D-Var, EnKF, EnKS, IEnKS, etc.

Combined data assimilation and machine learning

Second step: machine learning

Next, once a state trajectory $\mathbf{x}^\star_{0:K}$ has been obtained from data assimilation, one can focus on the machine learning problem and specifically $\boldsymbol{\omega}$:

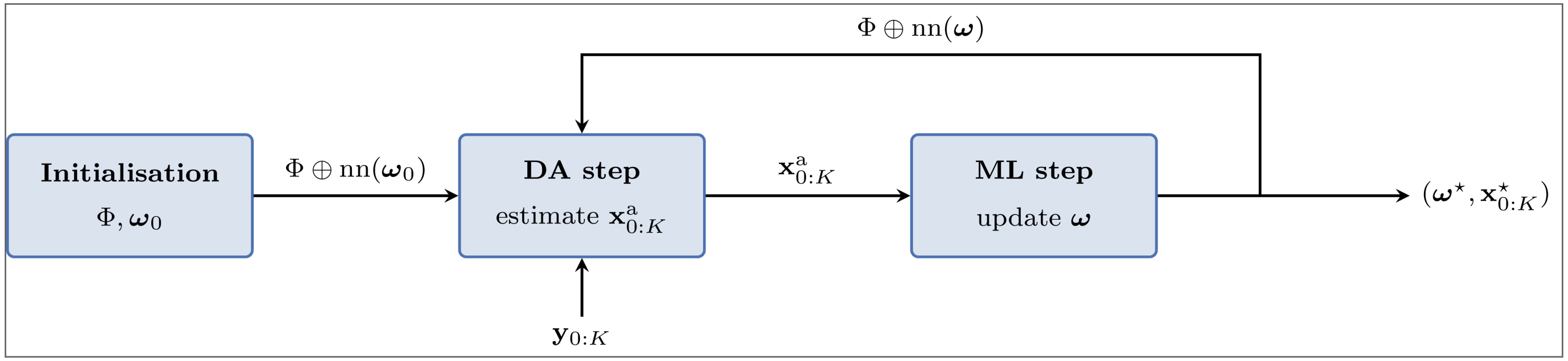
$$J(\boldsymbol{\omega}, \mathbf{x}^\star_{0:K}) = -J_\mathrm{b}(\mathbf{x}^\star_0) + \sum_{k=0}^{K} \|\mathbf{y}_k - H_k(\mathbf{x}^\star_k)\|^2_{\mathbf{R}_k^{-1}} + \sum_{k=1}^{K} \|\mathbf{x}^\star_k - \Phi(\mathbf{x}^\star_{k-1}) \\ - \mathrm{nn}(\boldsymbol{\omega}, \mathbf{x}^\star_{k-1})\|^2_{\mathbf{Q}_k^{-1}}.$$

The output is a learned neural net $\mathrm{nn}(\boldsymbol{\omega}^\star)$, and hence a correction of the hybrid physical/statistical model.

Combined data assimilation and machine learning

Coordinate descent

Moreover, it can be shown that one can iterate those two steps, which is known as a coordinate descent, following the schematic:



$\Phi \oplus \mathrm{nn}(\boldsymbol{\omega})$
Initialisation
$\Phi, \boldsymbol{\omega}_0$
$\Phi \oplus \mathrm{nn}(\boldsymbol{\omega}_0)$
DA step
estimate $\mathbf{x}^{\mathrm{a}}_{0:K}$
$\mathbf{x}^{\mathrm{a}}_{0:K}$
ML step
update $\boldsymbol{\omega}$
$(\boldsymbol{\omega}^\star, \mathbf{x}^\star_{0:K})$
$\mathbf{y}_{0:K}$

Fig.1 - Coordinate descent schematic for DAML.