Bayesian Deep Learning for Data Assimilation

Peter Jan van Leeuwen, borrowing ideas from discussions with many…

European Research Council

Established by the European Commission

Uncertainty Quantification in data assimilation

Since its embedding in Bayes Theorem data assimilation has a fairly complete way to describe and handle uncertainties. We distinguish:

1. Uncertainties in the prior of the state, using a nontrivial prior pdf
2. Uncertainties related to the measurement process, including representation errors
3. Uncertainties in the model equations
4. Leading to uncertainties in the posterior

This talk is about how machine learning can handle UQ and how DA and DL can be combined in a principled way.

UQ in Deep Learning

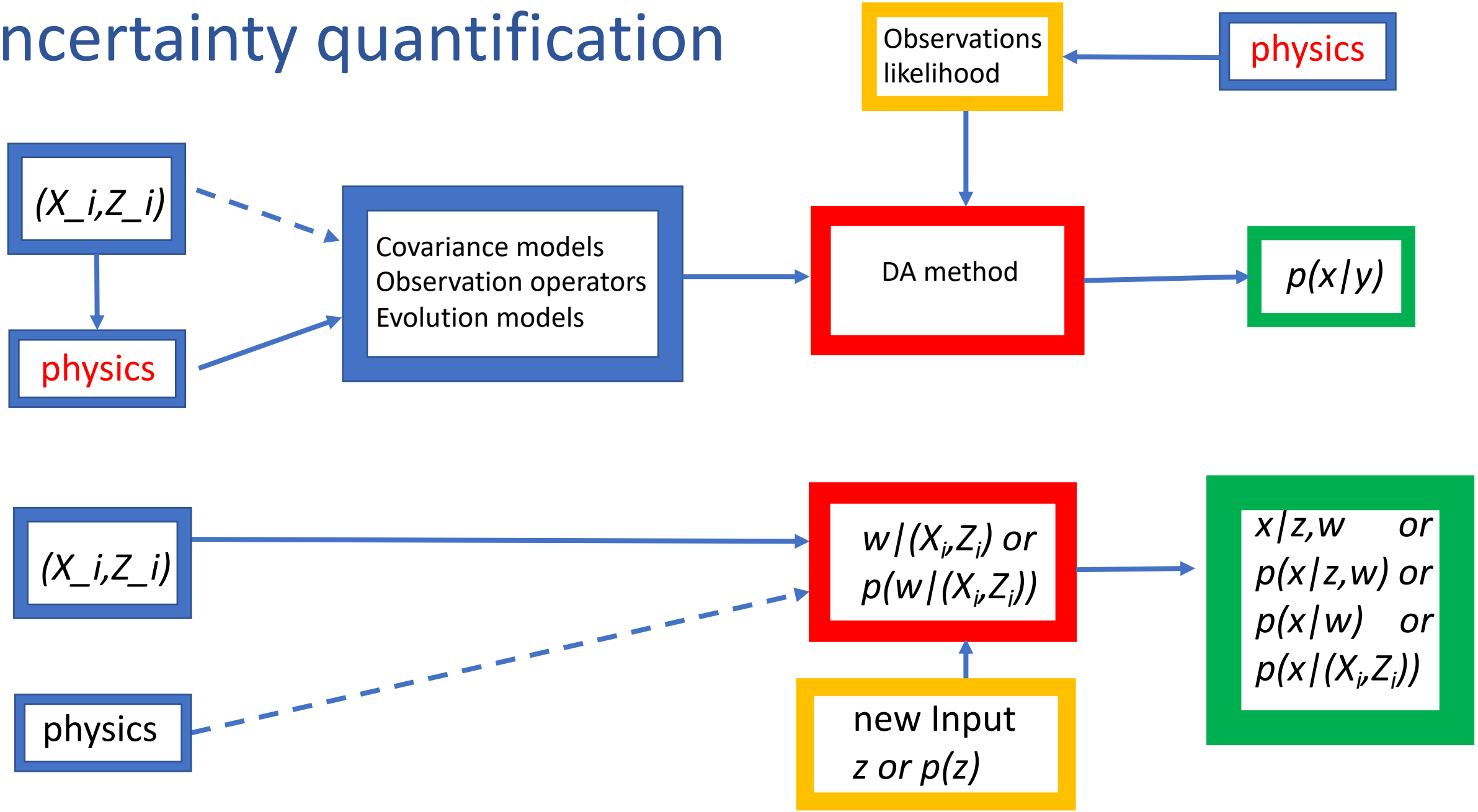
Since we need to provide predictions for longer lead times which (deep) machine learning can not handle well we assume that the input and output space of the machine are of the same dimension.

Machine learning started off as follows:

1. Prior on the weights (not input state), very simplistic, Tikhonov-like…
2. Uncertainties in the input-output pairs as a scaled identity matrix
3. ~~Uncertainties in the model equations (the weights)~~
4. ~~Uncertainties in the posterior~~

Much development is/was needed..

Uncertainty quantification



UQ for Deep Learning

The uncertainty sources in machine learning are

1) Uncertainty in the input-output pair relation used for training
2) Uncertainty in the new input
3) Uncertainty in the model (the neural network weights)
4) Leading to uncertainty in the posterior state

We will treat them one by one.

Uncertainty in the input-output pair relation

This corresponds to the likelihood in Data Assimilation.

Indeed, when setting up the costfunction for the NN we should use the 'likelihood' of the input-output relation

$$J(w) = -\log p(X_i|Z_i) - \log p(w)$$

but note that X_i is the output state of the NN, and Z_i is the input. The analogy with Data Assimilation is that the NN provides the observation operator.

As an example, if the relation has Gaussian error with covariance C the costfunction becomes

$$J(w) = \frac{1}{2}(X_i - Z_i)^T C^{-1}(X_i - Z_i) - \log p(w)$$

All these term have real meaning. One could include physics in C or add extra physical constraints.

Uncertainty in the NN

Deep learning needs to find the probability of state *x* given a set of climatological input-output pairs (X_i, Z_i).

DL provides an interface between climatology and the output:

$$p(x|(X_i, Z_i)) = \int p(x, w|(X_i, Z_i))\, dw$$

$$= \int p(x|w, (X_i, Z_i))p(w|(X_i, Z_i))\, dw$$

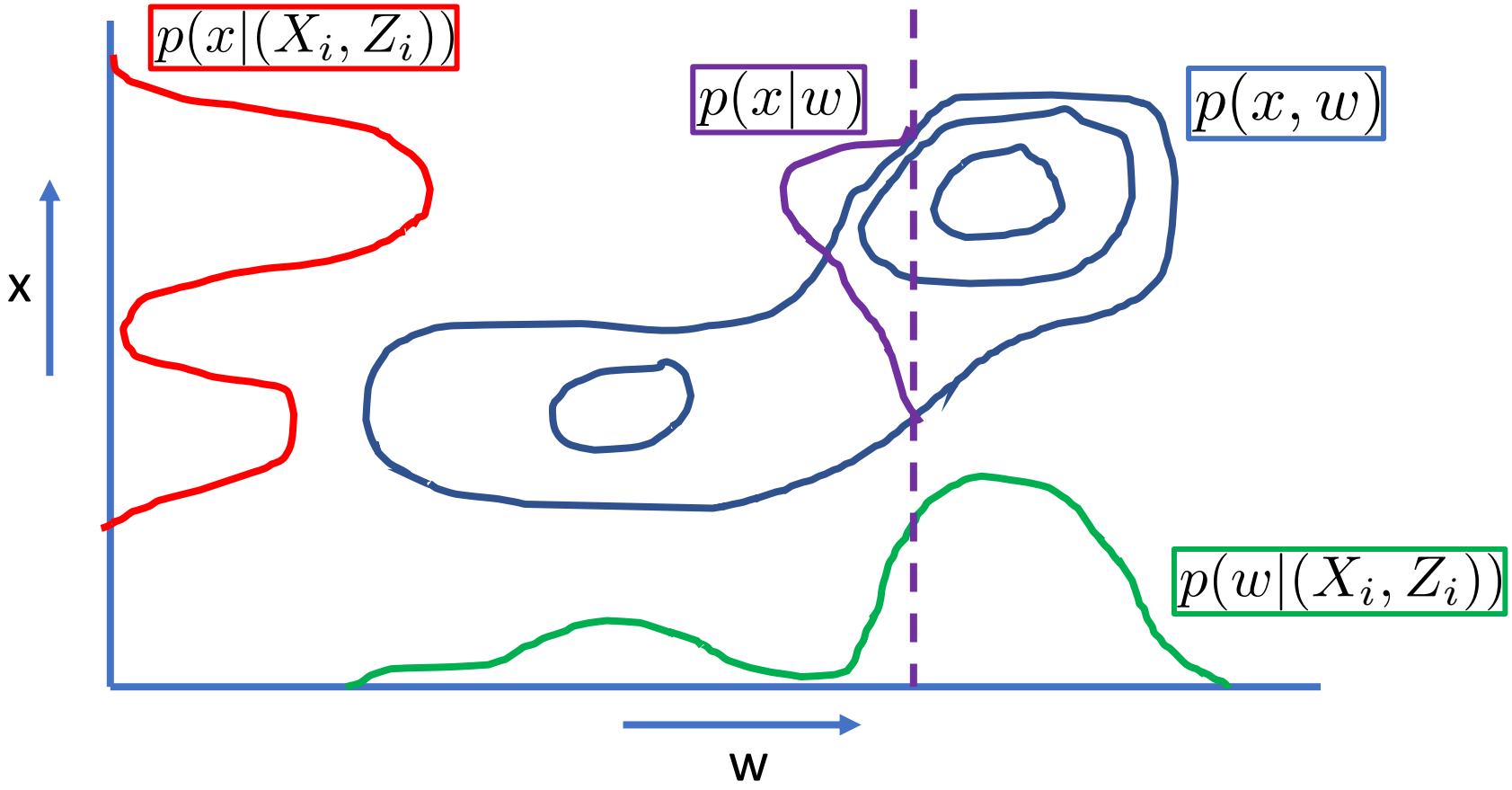
$$= \int p(x|w)p(w|(X_i, Z_i))\, dw$$

Note that $p(w|(X_i, Z_i))$ has many modes and is hard to find in general.

But we don't need the pdf itself, we just need to be able to sample from it!

Graphical illustration

We found $$p(x|(X_i, Z_i)) = \int p(x|w)p(w|(X_i, Z_i))\, dw$$



Note that *p(x|w)* is running the NN, *and* determining the uncertainty in the output.

Uncertainty in the input state

Now bring in the uncertainty in the input state *z,* in a similar manner:

$$p(x|(X_i, Z_i)) = \int p(x|w)p(w|(X_i, Z_i))\, dw = \int p(x, z|w)p(w|(X_i, Z_i))\, dwdz$$
$$= \int p(x|z, w)p(z|w)p(w|(X_i, Z_i))\, dwdz$$
$$= \int p(x|z, w)p(w|(X_i, Z_i))p(z)\, dwdz$$

The prior *p(z)* reflects our uncertainty in the input z, which can be a set of observations, or a model forecast, or indeed the output of another NN.

The complete UQ equations

A practical scheme would be to draw samples of NN's, so of the posterior of the weights (see later):

$$p(w|(X_i, Z_i)) = \frac{1}{M}\sum_{j=1}^{M}\delta(w - w_j)$$

leading to

$$p(x|(X_i, Z_i)) = \frac{1}{M}\sum_{j=1}^{M}\int p(x|z, w_j)p(z)\ dz$$

Hence the pdf of the new output state *x* given a set of input-output samples is a sum of

1. the pdf of the state conditioned on each set of weights and a possible input *z*
2. integrated over all possible values of that input weighted by p(z)

The above applies to all DNN, such as MLP, RNN, LSTM, GRU (Gated Recurrent Units)

We now look at how a few proposed NN methods that include UQ map onto this.

Sampling posterior weights + output uncertainty

Sample $p(w|(X_i, Z_i))$ by training *M* neural networks each using

1) different initial weights and

2) using a different order of the sample data

Giving us $$p(w|(X_i, Z_i)) = \frac{1}{M}\sum_{j=1}^{M}\delta(w - w_j)$$

The only part that is missing is $p(x|z, w_j)$. One solution is to:

1) From the testing data generate an approximation of $p(\epsilon_j(z))$ in which $\epsilon_j(z)$ is defined from $$x = f_{w_j}(z) + \epsilon_j(z)$$

2) We can now identify $p(x|z, w_j) = p(\epsilon_j(z))$

Deep Ensembles

B. Lakshminarayanan A. Pritzel C. Blundell (2017) arXiv:1612.01474v3

Deep Ensembles try to find $p(x|z,(X_i,Z_i))$ directly by assuming a shape for this pdf, e.g. a Gaussian or a Gaussian mixture. The steps are:

1) Train *M* neural networks to minimize

$$J(w) = -\log p_\theta(x|z,(X_i,Z_i)) \qquad = \frac{1}{2}\log\sigma^2(Z_i) + \frac{1}{2\sigma^2(Z_i)}(X_i-\mu(Z_i)^2$$

where each uses *different initial weights and a different order of the sample data*.

2) Average either the parameter vectors θ_j or the $p(x|z,w_j)$ to find $p(x|z,(X_i,Z_i))$

Notes 1) a standard MSE costfunction finds the weights, and the mean of $p(x|z,(X_i,Z_i))$

2) we could add a **prior on the parameters**

3) For NWP the parameter space is huge.

4) What is still missing is the uncertainty in z itself.

Bayesian Deep Learning

Bayesian Deep learning does the **inference on the weights** of the NN:

1. Start with a prior on the weights $p(w)$.
2. Perform training to infer posterior on the weights $p(w|(X_i, Z_i))$
3. This weights posterior is then used to derive a posterior pdf on any input state.
4. Since the number of weights is very large inference on them is impractical.
5. Many approximations exist: Laplace ('Gaussian'), variational Bayes, MCMC, Hybrid MC,..

Issues: 1) Computationally demanding, 2)Training is hard, many modes! 3) How to find a meaningful prior on the weights as they have no connection to the physics? 4) The other uncertainties are not addressed.

The point is that we do not need to have an accurate representation of the posterior of the weights as we will marginalize over them.

Does MC Drop Out provide uncertainty estimates ?

- Drop Out is a standard technique used to avoid overfitting.
- It consists of setting a randomly chosen set of weights equal to zero, a different set for each input-output pair during training. This defines the prior on the weights $p(w)$. It does not determine a posterior on the weights.
- In Monte-Carlo Drop Out the same NN is run several times on one new input using randomly chosen drop-out weight sets as samples from $p(w|(X_i, Z_i)$ and then use

$$p(x|z, (X_i, Z_i)) = \int p(x|z, w)p(w|(X_i, Z_i)\, dw$$

- However, there is no principled way to determine this MC sampling of the drop out weights, so the sampled NN's are not samples from $p(w|(X_i, Z_i)$!
- Furthermore, more sampling does not reduce the uncertainty.
- Experiment show that MC-Drop Out typically leads to 'uncertainty estimates' that are way too small.

Example of using DL in DA

Representation error: uncertain observation operator *H(x).* We can write:

$$p(x|y) = \frac{p(x)}{p(y)} p(y|x) = \frac{p(x)}{p(y)} \int p(y, H|x)\, dH$$

Now use

$$\int p(y, H|x)\, dH = \int p(y|H, x) p(H|x)\, dH$$

and determine $p(H|x)$ via Deep Learning:

$$\begin{aligned} p(H|x, (X_i, Z_i)) &= \int p(H, w|x, (X_i, Z_i))\, dw \\ &= \int p(H|x, w, (X_i, Z_i)) p(w|x, (X_i, Z_i))\, dx \\ &= \int p(H|x, w) p(w|(X_i, Z_i))\, dw \end{aligned}$$

Example of using DL in DA II

Use an ensemble of DL machines, e.g. different sample order order and initial weights:

$$p(H|x) = \int p(H|x,w)p(w|(X_i,Z_i))\,dw \approx \frac{1}{M}\sum_{i=1}^{M} p(H|x,w_i)$$

As an example, assume $p(H|x)$ is Gaussian with mean *h(x)* and variance *S,* from fitting to the NN result.
Furthermore, assume that the likelihood is Gaussian *N(H(x), R).* Then we find, performing the integral over the product of Gaussians:

$$p(y|x) = \int p(y|x,H)p(H|x)\,dH = N(h(x), R+S)$$

where *h(x)* and *S* are obtained from the NN.

'Any' component in the DA system can be treated via DL in this way.

Conclusions

- Uncertainty quantification for Deep Learning can be developed
- It can also be made practical via a small number of DL machines with different weights
- DL is similar to data assimilation in which climatology is given a dominant role.
- DL with unicertainty quantification can easily be implemented in existing linearized DA algorithms when Gaussian assumptions are made on the **output of the DL machines** (not on their weights).
- These Gaussian assumptions are not needed for non-Gaussian DA.