Data assimilation meets machine learning: Ensemble Kalman Inversion for neural-network training

Dmitry I. Kabanov

16 July 2020

Guest lecture for the Data Assimilation class

Introduction to machine learning

What is Machine Learning?

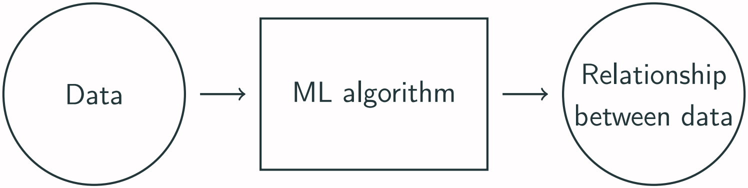
Machine-learning algorithms use statistics to find patterns in massive amounts of data. And data, here, encompasses a lot of things—numbers, words, images, clicks, what have you.

[K. Hao, MIT Technology Review, 2018]

What is Machine Learning?

Machine-learning algorithms use statistics to find patterns in massive amounts of data. And data, here, encompasses a lot of things—numbers, words, images, clicks, what have you.

[K. Hao, MIT Technology Review, 2018]



Process of finding the relationship between the data is called *learning*.

Learning process is minimization process

Learning process = minimization of a cost over a given set of mappings for given data:

$$\min_{u \in \mathcal{U}} \int_{\mathcal{X} \times \mathcal{Y}} \mathcal{L}(\mathcal{G}(x, u), y) \, \mathrm{d}\mathbb{P}(x, y)$$

where

- $x \in \mathcal{X}$ — independent variables (*features*)
- $y \in \mathcal{Y}$ — dependent variables (*labels*)
- $\mathcal{G}(x, u) : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ — model parameterized by $u \in \mathcal{U}$
- $\mathcal{L}(\mathcal{G}(x, u), y)$ — loss function that behaves like a metric between data y and the model $\mathcal{G}$

Learning process is minimization process

Learning process = minimization of a cost over a given set of mappings for given data:

$$\min_{u\in\mathcal{U}} \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\mathcal{G}(x,u),\, y)\,\mathrm{d}\mathbb{P}(x,y)$$

where

- $x \in \mathcal{X}$ — independent variables (*features*)
- $y \in \mathcal{Y}$ — dependent variables (*labels*)
- $\mathcal{G}(x,u) : \mathcal{X} \times \mathcal{U} \to \mathcal{Y}$ — model parameterized by $u \in \mathcal{U}$
- $\mathcal{L}(\mathcal{G}(x,u),\, y)$ — loss function that behaves like a metric between data y and the model $\mathcal{G}$

This minimization problem is not necessarily well defined and often requires adding regularizations to be able to find a solution.

Learning process with finite datasets

Learning process = minimization of a loss over a given set of mappings for given data:

$$\min \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\mathcal{G}(x,u),\, y)\, \mathrm{d}\mathbb{P}(x,y)$$

In practice, $\mathbb{P}(x,y)$ is not given and also we have only finite dataset:

$$(\mathrm{x},\mathrm{y}) = \{(x_i, y_i)\},\ i = 1,\ldots,N$$

Examples of ML problems: Boston housing price prediction

Dataset consists of 506 elements and aggregates information about Boston's suburbs using 14 features (input variables), such as:

- per capita crime rate by town (real)
- average number of room (integer)
- weighted distance to five Boston employment centers (real)
- ratio of pupils to teachers (real)

and the output value is the median price of occupied houses. The goal is to find mapping that will predict the price of a given house.

Examples of ML problems: Boston housing price prediction

Dataset consists of 506 elements and aggregates information about Boston's suburbs using 14 features (input variables), such as:

- per capita crime rate by town (real)
- average number of room (integer)
- weighted distance to five Boston employment centers (real)
- ratio of pupils to teachers (real)

and the output value is the median price of occupied houses. The goal is to find mapping that will predict the price of a given house.

- $\mathcal{X} \subset \mathbb{R}^{14}$
- $\mathcal{Y} \subset \mathbb{R}$
- $N = 506$

Examples of ML problems: Clothing classification

Fashion MNIST dataset consists of 60 000 entries:

- Each entry as grayscale 28×28 image
- Each images contains a piece of clothing from 10 classes: T-shirt, pullover, dress, and so on.

[https://github.com/zalandoresearch/fashion-mnist]

Examples of ML problems: Clothing classification

Fashion MNIST dataset consists of 60 000 entries:

- Each entry as grayscale 28×28 image
- Each images contains a piece of clothing from 10 classes: T-shirt, pullover, dress, and so on.

[https://github.com/zalandoresearch/fashion-mnist]

- $\mathcal{X} \subset \mathbb{R}^{28 \times 28 = 784}$
- $\mathcal{Y} \subset \mathbb{P}^{10}$
- $N = 60000$

Supervised learning

Learning process = minimization of a cost over a given set of mappings for given data:

$$\min \int_{\mathcal{X}\times\mathcal{Y}} \mathcal{L}(\mathcal{G}(x,u), y)\,\mathrm{d}\mathbb{P}(x,y)$$

Suppose that the dataset (x,y) is i.i.d from $\mathbb{P}(x,y)$. Then we replace integral with the Monte Carlo approximation

$$\arg\min_{u\in\mathcal{U}} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\mathcal{G}(x_i;u), y_i\right) + R(u)$$

and also add regularizer $R(u) : \mathcal{U} \to \mathcal{Y}$ that helps with solving the minimization problem.

Popular choice is $R(u) = \lambda \left\|u\right\|_2^2$ for positive λ.

Supervised learning: applications

Supervised learning is the most common type of machine learning.

Uses are:

- image classification
- natural language processing
- object detection

Semi-supervised learning

The dataset is $\mathrm{x} = (x_j)$, $j \in \mathcal{Z}$, while $\mathrm{y} = (y_j)$, $j \in \mathcal{Z}'$, where $\mathcal{Z}' \in \mathcal{Z}$ with $|\mathcal{Z}'| \ll |\mathcal{Z}|$.

Then learning is minimization problem

$$\arg\min_{u \in \mathcal{U}} \frac{1}{|\mathcal{Z}'|} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{G}(x_i; u), y_i\right) + R(\mathrm{x}; u)$$

Notice that misfit term depends only on labeled data with indices in $\mathcal{Z}'$, while regularizer term depends on labeled+unlabeled data.

Applications include problems where it is easy to collect input data but difficult to obtain output data (for example, in medicine, it is easier/cheaper to produce MR images then to have correct diagnosis for them).

Online learning

Elements of the dataset are given sequentially, point by point. With each element we can improve the estimate of parameter u.

To simplify it even further, assume that the process is markovian and improvement is based on using new data element and previous estimate of u:

$$\arg\min_{u \in \mathcal{U}} \mathcal{L}\left(\mathcal{G}(x_j; u), y_j\right) + R(\mathrm{x}; u_{j-1})$$

This type of learning is just supervised learning with cheaper computational costs. Also, it should be used when data acquisition is sequential by nature (distributed in time).

Learning is an inverse problem

Every learning process can be interpreted as an inverse problem of finding parameter $u \in \mathcal{U}$ from given finite dataset (x_i, y_i), $i = 1, \dots, N$.

Inverse problem is

$$\mathrm{y} = \mathrm{G}(u|x) + \eta$$

where $\mathrm{G}(u|\mathrm{x}) = [\mathcal{G}(x_1, u), \dots, \mathcal{G}(x_N, u)]^\mathsf{T}$ is the concatenation of the model evaluated at each data element, and $\eta \sim \pi$ is a $\mathcal{Y}^N$ random variable that models noise in the data.

Then Bayes' theorem states that

$$\pi(u|\mathrm{x}, \mathrm{y}) \propto \pi(\mathrm{x}, \mathrm{y}|u)\pi_0(u)$$

where $\pi_0(u)$ is the prior distribution of u.

Learning is an inverse problem

Assume that all elements in the dataset are independent of each other.

Then taking minus logarithm of the Bayes' formula, we arrive at

$$\pi(u|\mathrm{x}, \mathrm{y}) \propto \sum_{i=1}^{N} \mathcal{L}(\mathcal{G}(u, x_i), y_i) + R(u)$$

where

$$\pi(\mathrm{x}, \mathrm{y}|u) \propto \sum_{i=1}^{N} \mathcal{L}(\mathcal{G}(u, x_i), y_i)$$

and

$$\pi_0(u) \propto R(u)$$

Learning is an inverse problem

Assume that all elements in the dataset are independent of each other.

Then taking minus logarithm of the Bayes' formula, we arrive at

$$\pi(u|\mathrm{x},\mathrm{y}) \propto \sum_{i=1}^{N} \mathcal{L}(\mathcal{G}(u,x_i),y_i) + R(u)$$

where

$$\pi(\mathrm{x},\mathrm{y}|u) \propto \sum_{i=1}^{N} \mathcal{L}(\mathcal{G}(u,x_i),y_i)$$

and

$$\pi_0(u) \propto R(u)$$

Therefore, the learning process can be interpreted as finding *maximum a posteriori* (MAP) estimate of u.

Feed-forward neural networks

Let $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^m$.

Then feed-forward neural network with L layers is a mapping from $\mathcal{X}$ to $\mathcal{Y}$ of the following compositional structure:

$$\mathcal{G}(x; u) = S \circ A \circ F_{L-1} \circ \cdots \circ F_1(x)$$

where

- $S : \mathbb{R}^m \to V \subseteq \mathbb{R}^m$, where usually $V = \mathbb{R}^m$ for regression and $V = \mathbb{P}^m$ for classification
- A is an affine map, which is also called **output layer**
- $F_j : \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}$ nonlinear continuous functions, with $n_0 = n$, which are called **hidden layers**

Feed-forward neural networks

Let $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y} = \mathbb{R}^m$.

Then feed-forward neural network with L layers is a mapping from $\mathcal{X}$ to $\mathcal{Y}$ of the following compositional structure:

$$\mathcal{G}(x; u) = S \circ A \circ F_{L-1} \circ \cdots \circ F_1(x)$$

where

- $S : \mathbb{R}^m \to V \subseteq \mathbb{R}^m$, where usually $V = \mathbb{R}^m$ for regression and $V = \mathbb{P}^m$ for classification
- A is an affine map, which is also called **output layer**
- $F_j : \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}$ nonlinear continuous functions, with $n_0 = n$, which are called **hidden layers**

This neural networks are also called networks with $L - 1$ hidden layers.

Fully-connected NNs are subtype of feed-forward NNs

If for feed-forward neural network

$$\mathcal{G}(x;u) = S \circ A \circ F_{L-1} \circ \cdots \circ F_1(x)$$

we specialize hidden layers to pair "nonlinearity + affine map":

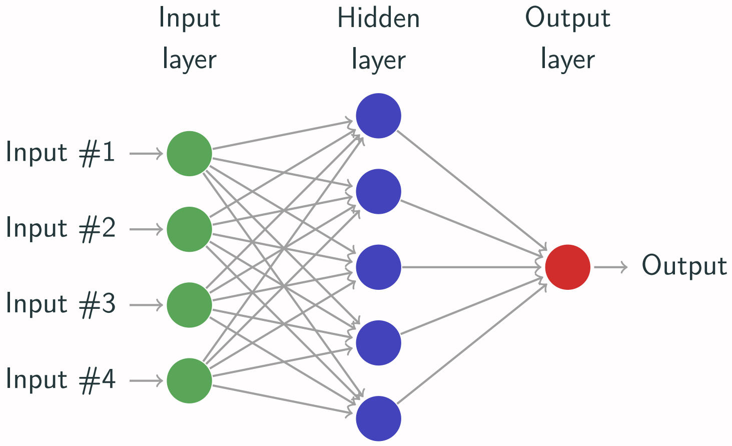
$$F_j(x_{j-1}) = \sigma(W_j z_{j-1} + b_j)$$

where

- $W_j \in \mathbb{R}^{n_{j-1}} \times \mathbb{R}^{n_j}$ are **weights** matrices
- $b_j \in \mathbb{R}^{n_j}$ are **bias** vectors
- nonlinear function σ is applied componentwise

then we obtain **fully-connected** or **dense** neural network, which is also called **multi-layer perceptron**.

Visualization of multilayer perceptron $\mathbb{R}^4 \to \mathbb{R}$ with one hidden layer



[`https://texample.net/tikz/examples/neural-network/`, with modifications]

Training process and algorithms

There are two most commonly used loss functions

For regression – least squares error

$$\mathcal{L}(y, y') = \|y - y'\|_{\mathcal{Y}}^2$$

is used, which comes from additive Gaussian noise model.

There are two most commonly used loss functions

For regression – least squares error

$$\mathcal{L}(y, y') = \|y - y'\|^2_{\mathcal{Y}}$$

is used, which comes from additive Gaussian noise model.

For classification – Shannon's cross-entropy

$$\mathcal{L}(y, y') = -\langle y, \log y' \rangle_{\mathcal{Y}}$$

is used. Notice that cross-entropy is difficult to interpret in a Bayesian sense as $\mathcal{L}$ does not depend on $y - y'$, hence, cannot be expressed in terms of the noise probability distribution.

Gradient descent

Let assume that sought-for parameter vector u is a smooth function of time: $u : [0; +\infty) \to \mathcal{U}$.

Also, let $\Phi(u; \mathrm{x}, \mathrm{y})$ denote loss function along with regularizer.

Then the search for u that minimizes Φ is governed by gradient descent equation with initial condition

$$\dot{u} = -\nabla_u \Phi(u; \mathrm{x}, \mathrm{y}), \quad u(0) = u_0.$$

Gradient descent

Let assume that sought-for parameter vector u is a smooth function of time: $u : [0; +\infty) \to \mathcal{U}$.

Also, let $\Phi(u; \mathrm{x}, \mathrm{y})$ denote loss function along with regularizer.

Then the search for u that minimizes Φ is governed by gradient descent equation with initial condition

$$\dot{u} = -\nabla_u \Phi(u; \mathrm{x}, \mathrm{y}), \quad u(0) = u_0.$$

In practice, this equation is solved using forward Euler method

$$u_{n+1} = u_n - h \nabla_u \Phi(u_n; \mathrm{x}, \mathrm{y})$$

where $h \in \mathbb{R}$ is a time step, which is called **learning rate** in ML parlance. Often h is decreasing function of the iteration.

In practice, mini-batch stochastic gradient descent is used.

For a large dataset ($N \gg 1$), all data is split into mini-batches that are chosen at random and have sizes $\beta \in \mathbb{N}$ or $N \mod \beta$.

```
Given: $u_0$, $n = 0$, $E$, $\beta$, $K = \lceil N/\beta \rceil$
for $e \in (1, \ldots, E)$ do
    draw $(B_1, \ldots, B_K) \sim \mathrm{Unif}(1, N)$ w/o replacement;
    for $b \in (B_1, \ldots, B_K)$ do
        $u_n + 1 = u_n - h\nabla_u \Phi(u; b, \mathrm{x}, \mathrm{y})$ ;
    end
end
```

$$\Phi(u; b, \mathrm{x}, \mathrm{y}) = \frac{1}{|b|} \sum_{i \in b} \mathcal{L}(\mathcal{G}(x_j; u), y_j) + R(u)$$

In practice, mini-batch stochastic gradient descent is used.

For a large dataset ($N \gg 1$), all data is split into mini-batches that are chosen at random and have sizes $\beta \in \mathbb{N}$ or $N \mod \beta$.

```
Given: $u_0$, $n = 0$, $E$, $\beta$, $K = \lceil N/\beta \rceil$
for $e \in (1, \dots, E)$ do
    draw $(B_1, \dots, B_K) \sim \mathrm{Unif}(1, N)$ w/o replacement;
    for $b \in (B_1, \dots, B_K)$ do
        $u_n + 1 = u_n - h \nabla_u \Phi(u; b, \mathrm{x}, \mathrm{y})$ ;
    end
end
```

inner loop is called **epoch**

$$\Phi(u; b, \mathrm{x}, \mathrm{y}) = \frac{1}{|b|} \sum_{i \in b} \mathcal{L}(\mathcal{G}(x_j; u), y_j) + R(u)$$

Ensemble Kalman Inversion

For a given dataset (x, y), consider general inverse problem

$$\mathrm{y} = \mathrm{G}(u) + \eta$$

where $\eta \sim N(0, \Gamma)$ and x is absorbed into definition of G.

Let $\{u^{(j)}\}_{j=1}^{J} \in \mathcal{U}$ be ensemble of estimates of u, with initial estimates drawn from prior distribution. To simplify notation, let u^j mean $u^{(j)}$.

This ensemble evolve in time according to the EKI dynamic:

$$\dot{u}^j = -C^{uw}(u)\,\Gamma^{-1}\left(G(u^j) - y\right) \tag{1}$$

$$u^j(0) = u_0^j \tag{2}$$

for $j = 1, \ldots, J$.

Ensemble Kalman inversion

$$\dot{u}^j = -C^{uw}(u)\,\Gamma^{-1}\left(G(u^j) - y\right)$$
$$u^j(0) = u_0^j$$

where empirical covariance is

$$C^{uw}(u) = \frac{1}{J}\sum_{j=1}^{J}\left(u^j - \bar{u}\right) \otimes \left(G(u^j) - \bar{G}\right)$$

and the means are

$$\bar{u} = \frac{1}{J}\sum_{k=1}^{J} u^k, \quad \bar{G} = \frac{1}{J}\sum_{k=1}^{J} G(u^k)$$

Derivation of the Ensemble Kalman Inversion

To relate EnKF to the general inverse problem with one observation

$$y = G(u) + \eta$$

consider artificial time dynamics:

$$\begin{aligned} u_{n+1} &= u_n \\ y_{n+1} &= G(u_{n+1}) + \eta_{n+1} \end{aligned}$$

Now, add auxiliary variable w:

$$\begin{aligned} u_{n+1} &= u_n \\ w_{n+1} &= G(u_{n+1}) \\ y_{n+1} &= w_{n+1} + \eta_{n+1} \end{aligned}$$

Derivation of the Ensemble Kalman Inversion

Let $v = (u, w)^\mathsf{T}$ and $\Psi = (u, \mathrm{G}(u))^\mathsf{T}$.

Introduce observation operator $H = [0, I]$ and $H^\perp = [I, 0]^\mathsf{T}$ such that $Hv = u$ and $H^\perp v = w$.

Then the problem can be formulated in the standard data-assimilation setting:

$$
\begin{aligned}
v_{n+1} &= \Psi(v_n) \\
y_{n+1} &= Hv_{n+1} + \eta_{n+1}
\end{aligned}
$$

Derivation of the Ensemble Kalman Inversion

For the problem

$$v_{n+1} = \Psi(v_n)$$
$$y_{n+1} = Hv_{n+1} + \eta_{n+1}$$

Ensemble Kalman filter can be applied for ensemble of size J

$$\widehat{v}^j_{n+1} = \Psi(v^j_n) \qquad \bar{v}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} \widehat{v}^j_{n+1}$$

$$\widehat{C}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} \left(\widehat{v}^j_{n+1} - \bar{v}_{n+1}\right) \otimes \left(\widehat{v}^j_{n+1} - \bar{v}_{n+1}\right)$$
$$v^j_{n+1} = \widehat{v}^j_{n+1} + K_{n+1}\left(y^j_{n+1} - H\widehat{v}^j_{n+1}\right)$$

with Kalman gain

$$K_{n+1} = \widehat{C}_n H^\mathsf{T}\left(H\widehat{C}_n H^\mathsf{T} + \Gamma\right)^{-1}$$

Formulas can be simplified for this particular problem

Recall that Then

$$\widehat{C}_n = \begin{bmatrix} C^{uu}_{n+1} & C^{uw}_{n+1} \\ \left(C^{uw}_n\right)^{\mathsf{T}} & C^{ww}_{n+1} \end{bmatrix} \qquad \bar{v}_{n+1} = \begin{pmatrix} \bar{u}_{n+1} \\ \bar{w}_{n+1} \end{pmatrix}$$

with

$$\bar{u}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} u_n^j \qquad \bar{w}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} G\left(u_n^j\right) := \bar{G}_n$$

$$\bar{C}^{uw}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} (u_n - \bar{u}_{n+1}) \otimes \left(G(u_n^j) - \bar{G}_n\right)$$

$$\bar{C}^{ww}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} \left(G(u_n^j) - \bar{G}_n\right) \otimes \left(G(u_n^j) - \bar{G}_n\right)$$

Formulas can be simplified for this particular problem

As $H = [0, I]$, then Kalman gain

$$K_{n+1} = \widehat{C}_{n+1} H^{\mathsf{T}} \left(H \widehat{C}_{n+1} H^{\mathsf{T}} + \Gamma \right)^{-1}$$

simplifies to

$$K_{n+1} = \begin{bmatrix} C_{n+1}^{uw} \left(C_{n+1}^{ww} + \Gamma \right)^{-1} \\ C_{n+1}^{ww} \left(C_{n+1}^{ww} + \Gamma \right)^{-1} \end{bmatrix}$$

and due to

$$u_{n+1} = H^{\perp} v = [I, 0] v$$

update step for u is

$$u_{n+1} = u_n^j + C_{n+1}^{uw} \left(C_{n+1}^{ww} + \Gamma \right)^{-1} \left(y_{n+1}^j - G(u_n^j) \right)$$

Algorithm for Ensemble Kalman Inversion

```
Given  : Prior distribution $\pi_0$, observations $Y_N$, ensemble size $J$
Init   : Draw $J$ particles $u_0^j \sim \pi_0$
for $n = 0, \dots, N-1$ do
    Compute $\bar{u}_{n+1} = \frac{1}{J}\sum_{j=1}^{J} u_n^j$ ;
    Compute $\bar{G}_n = \frac{1}{J}\sum_{j=1}^{J} G(u_n^j)$ ;
    Compute $\bar{C}_{n+1}^{uw} = \frac{1}{J}\sum_{j=1}^{J} (u_n - \bar{u}_{n+1}) \otimes \left(G(u_n^j) - \bar{G}_n\right)$ ;
    Compute $\bar{C}_{n+1}^{ww} = \frac{1}{J}\sum_{j=1}^{J} \left(G(u_n^j) - \bar{G}_n\right) \otimes \left(G(u_n^j) - \bar{G}_n\right)$ ;
    for $j = 1, \dots, J$ do
        $u_{n+1}^j = u_n^j + C_{n+1}^{uw} \left(C_{n+1}^{ww} + \Gamma\right)^{-1} \left(y_{n+1}^j - G(u_n^j)\right)$ ;
    end
end
Output: $J$ particles $u_N^1, \dots, u_N^J$
```

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -C^{uw}(u)\,\Gamma^{-1}\left(\mathrm{G}(u^j) - \mathrm{y}\right)$$

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -C^{uw}(u)\,\Gamma^{-1}\left(\mathrm{G}(u^j) - \mathrm{y}\right)$$

Substitute definition of $C^{uw}(u)$:

$$\dot{u}^j = -\frac{1}{J}\sum_{k=1}^{J}\left(u^k - \bar{u}\right)\left(\mathrm{G}(u^k) - \bar{\mathrm{G}}\right)^{\mathsf{T}}\Gamma^{-1}\left(\mathrm{G}(u^j) - \mathrm{y}\right)$$

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -C^{uw}(u)\,\Gamma^{-1}\left(\mathrm{G}(u^j) - \mathrm{y}\right)$$

Substitute definition of $C^{uw}(u)$:

$$\dot{u}^{\,j} = -\frac{1}{J}\sum_{k=1}^{J}\left(u^{\,k} - \bar{u}\right)\left(\mathrm{G}(u^{\,k}) - \bar{\mathrm{G}}\right)^{\mathsf{T}}\Gamma^{-1}\left(\mathrm{G}(u^{\,j}) - \mathrm{y}\right)$$

Let model be linear: $\mathrm{G}(u) = Au$. Then

$$\dot{u}^{\,j} = -\frac{1}{J}\sum_{k=1}^{J}\left(u^{\,k} - \bar{u}\right)\left(u^{\,k} - \bar{u}\right)^{\mathsf{T}} A^{\mathsf{T}}\,\Gamma^{-1}\left(Au^{\,j} - \mathrm{y}\right)$$

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -\frac{1}{J}\sum_{k=1}^{J}\left(u^k-\bar{u}\right)\left(u^k-\bar{u}\right)^{\mathsf{T}} A^{\mathsf{T}}\Gamma^{-1}\left(Au^j-\mathrm{y}\right)$$

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -\frac{1}{J}\sum_{k=1}^{J}\left(u^k - \bar{u}\right)\left(u^k - \bar{u}\right)^{\mathsf{T}} A^{\mathsf{T}}\Gamma^{-1}\left(Au^j - \mathrm{y}\right)$$

Introduce empirical covariance operator

$$C(u) = \frac{1}{J}\sum_{k=1}^{J}\left(u^k - \bar{u}\right)\left(u^k - \bar{u}\right)^{\mathsf{T}}$$

and loss function

$$\Phi(u;\mathrm{y}) = \frac{1}{2}\|\mathrm{y} - Au\|_{\Gamma}^{2}$$

Ensemble Kalman Inversion is derivative-free optimizer

$$\dot{u}^j = -\frac{1}{J}\sum_{k=1}^{J}\left(u^k-\bar{u}\right)\left(u^k-\bar{u}\right)^\top A^\top \Gamma^{-1}\left(Au^j - \mathrm{y}\right)$$

Introduce empirical covariance operator

$$C(u) = \frac{1}{J}\sum_{k=1}^{J}\left(u^k-\bar{u}\right)\left(u^k-\bar{u}\right)^\top$$

and loss function

$$\Phi(u;\mathrm{y}) = \frac{1}{2}\|\mathrm{y}-Au\|_\Gamma^2$$

Then

$$\dot{u}^j = -C(u)\,\nabla_u\Phi(u^j;\mathrm{y})$$

Ensemble Kalman Inversion and Gradient Descent are close to each other

Gradient Descent

$$\dot{u} = -\nabla_u \Phi(u; \mathrm{y})$$

Ensemble Kalman Inversion

$$\dot{u}^j = -C(u)\, \nabla_u \Phi(u^j; \mathrm{y})$$

Ensemble Kalman Inversion and Gradient Descent are close to each other

Gradient Descent	Ensemble Kalman Inversion
$\dot{u} = -\nabla_u \Phi(u; \mathrm{y})$	$\dot{u}^j = -C(u)\, \nabla_u \Phi(u^j; \mathrm{y})$

Ensemble Kalman Inversion can be viewed as Gradient Descent for each particle but gradient direction is corrected through the covariance matrix.

Thank you!