Fast approximation of complicated models
Mathematical foundations and application trends

February 28, 2017
Juan Pablo Carbajal

Eawag: Swiss Federal Institute of Aquatic Science and Technology
Background
We are interested in doing statistical inference on the dynamic state of a whole system based on a finite set of indirect measurements as well as estimating parameters related to the underlying process.

- **Methods:**
  - Regularized regression (RR)
  - Gaussian processes (GP)
  - Kalman filters (KF)

- **Unification:** RR $\iff$ GP $\iff$ KF

- **What’s an emulator?**

- **Tricks for large PDE models.**
Regularized Regression (RR): problem definition

Regularized regression with quadratic (convex) error: design data set of input-output values \( \{t_i, y_i\} = (T, y(T)) \) find \( f(t) \) that approximates the data and provides good predictions for unseen \( t \):

\[
\min_f \sum_{j=1}^{N} (y_j - f(t_j))^2 + \kappa \|Rf\|^2
\]

\( f(t) \) should be close to the design data, but it should be regular according to \( R \).
Regularized Regression (RR): problem definition

A $N + 1$-degree polynomial regression on $N$ points with

$$\| Rf \| = \int |\frac{d^2 f}{dt^2}|$$
Regularization operator

\[
\min_f \sum_{j=1}^{N} (y_j - f(t_j))^2 + \kappa \|Rf\|^2
\]

\[
J[f] = \sum_{j=1}^{N} \left( y_j - \int f(t)\delta(t-t_j)dt \right)^2 + \kappa \langle Rf(t), Rf(t) \rangle
\]

Searching for the critical point of that functional leads to

\[
R^\dagger Rf(t) = \sum_{j=1}^{N} \frac{y_j - f_j}{\kappa} \delta(t-t_j)
\]

\[
R^\dagger RG(t, t') = \delta(t-t'),
\]

---

Regularization operator

\[
\min_f \sum_{j=1}^{N} (y_j - f(t_j))^2 + \kappa \|Rf\|^2
\]

Solution

\[
f(t) = G(t, T) \left( G(T, T) + \kappa I \right)^{-1} (y(T) - n(T)) + n(t).
\]

Where \(G(t, t')\) is the Green’s function of the operator \(R^\dagger R\). If \(R\) has a Green’s function \(g(t, t')\) (and \(R^\dagger, g^\dagger(t, t')\)), then

\[
G(t, t') = \int g(t, u)g^\dagger(u, t')du
\]

---

A GP is a distribution of functions defined by a mean function \( m(t) \) and a covariance function \( k(t, t') \).

A prior GP conditioned on a design data set \( \{t_i, y_i\} = (T, y(T)) \), gives posterior GP.

The mean function of posterior GP is used for prediction and the formula reads

\[
\text{mean of posterior GP} = f(t) = k(t, T) \left( k(T, T) + \kappa I \right)^{-1} (y(T) - m(T)) + m(t).
\]

The evaluated covariance function \( k(T, T) \) is the covariance matrix.
1st order ODE with piece-wise constant input and random input, giving distribution of functions

\[ Lf(t) - u(t) = \eta(t) \]

\[ \eta(t) \sim \mathcal{N}(0, \Sigma\delta(t - t')) \]
Example: ODE

Covariance function
Example: ODE

\[ f(t) = k(t, t_i) \left( k(t_i, t_i) \right)^{-1} \hat{f}(t_i) + L^{-1} u(t) \]
Kalman filters (KF): characterization

An iterative method ($O(Tn^3)$) to predict (hidden) states of a $n$-state space dynamic model

$$\dot{x}(t) = Ax(t) + Bu(t) + \nu(t)$$
$$y(t_k) = Hx(t_k) + Du(t_k) + \epsilon(t_k)$$

$x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $u \in \mathbb{R}^l$

$\nu \sim GP(0, Q)$, $\epsilon \sim N(0, P)$

This is the continuous-time model discrete-time measurements flavor of KF. It can be extended to nonlinear (and non-Gaussian) systems: the Extended KF (model linearization), or the Unscented KF (posterior approximation).
Algorithm structure

Initial values

State update
1. Simulate forward: predict state.
2. Propagate state error covariance.

Measurement update
1. Compute Kalman gain.
2. Update state estimate with data.
3. Update state error covariance.
Methodological unification
Comparing GP and RR

**RR solution**

\[
f(t) = G(t, T) \left( G(T, T) + \kappa I \right)^{-1} (y(T) - n(T)) + n(t)
\]

**GP predictive mean**

\[
f(t) = k(t, T) \left( k(T, T) + \kappa I \right)^{-1} (y(T) - m(T)) + m(t).
\]
Relations between GP and RR

Gaussian process

\[ p_K(f) \]
zero mean

kernel function

\[ k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \]
pos. def.

covariance operator

\[ K : \mathcal{H} \rightarrow \mathcal{H} \]
sym. pos. def.

RKHS

\[ \| \cdot \|_K \]
inner prod.

regularization operator

\[ R : \mathcal{H} \rightarrow \mathcal{G} \]
one-to-one

---

Relations between GP and RR

"... all [reviewers] mention the affinity between my results ... and the Kalman filter, ... If only the work in these fields were more readily accessible to the statistician who (like me) is not a specialist pure mathematician in terms familiar to him, much duplication could be avoided. In fact I explicitly denied any originality for these result."³

Relations between GP and KF

- In Steinke & Schölkopf, GP and KF equivalence based on optimal estimation of solution.
- Inversion of dynamical system’s tridiagonal covariance matrices similar to KF\(^3\).
- Stationary covariance matrices can be converted to state-space models, where KF can be applied\(^4\), via Wiener-Khinchin theorem.


Relations between GP and KF

- In Steinke & Schölkopf, GP and KF equivalence based on optimal estimation of solution.
- Inversion of dynamical system’s tridiagonal covariance matrices similar to KF.
- Stationary covariance matrices can be converted to state-space models, where KF can be applied\(^3\), via Wiener-Khinchin theorem.

\[ \kappa(t - t') \propto \int S(\omega)e^{i\omega(t-t')}d\omega \]

Concretely

Spatio-temporal GP representation

$$f(r, t) \sim \mathcal{GP}(0, \kappa(r, t; r', t'))$$

$$y_k = \mathcal{H}_k f(r, t_k) + \epsilon_k$$

Linear stochastic partial differential equation

$$\frac{\partial f(r, t)}{\partial t} = \mathcal{F} f(r, t) + L w(r, t)$$

$$y_k = \mathcal{H}_k f(r, t_k) + \epsilon_k$$

$O((Tn)^3)$ general matrix inversion

$O(Tn^3)$ infinite-dimensional Kalman filtering and smoothing

1. Compute spectral density (SD) via Fourier transform of covariance
2. Approximate SD with a rational function (if necessary)
3. Find stable transfer function (poles in upper half complex plane)
4. Transform to SS using control theory methods
Visualize these GPs\textsuperscript{45}

\begin{enumerate}
\item Covariance function view
\item State space model view
\end{enumerate}


Example: isotropic Matérn covariance function

\[
\kappa(t - t') = \sigma^2 \lambda (t - t') K_1 [\lambda (t - t')]
\]

\[
\frac{\partial f(t)}{\partial t} = \left[\begin{array}{cc}
0 & 1 \\
-\lambda & -2\sqrt{\lambda}
\end{array}\right] f(t) + \left[\begin{array}{c}
0 \\
1
\end{array}\right] w(x, t)
\]

\[
r = \| (x, t) - (x', t') \|
\]

\[
\kappa(r) = \sigma^2 \lambda r K_1 (\lambda r)
\]

\[
\frac{\partial f(x, t)}{\partial t} = \left[\begin{array}{cc}
0 & 1 \\
\frac{\partial^2}{\partial x^2} - \lambda & -2\sqrt{\lambda} - \frac{\partial^2}{\partial x^2}
\end{array}\right] f(x, t) + \left[\begin{array}{c}
0 \\
1
\end{array}\right] w(x, t)
\]
Intermediate summary

- SS, GP and RR are unified (linear, stationary, convex loss-function): interpretability, algorithmic versatility
- GP \(\rightarrow\) SS allows for inference linear in the number of time samples
- RR \(\rightarrow\) GP allows non-parametric curve fitting
What is an emulator?
Emulation: Idea
TIMTOWTDI

What is an emulator?
Emulation: more formal

(a) $\mu = (0.2, 0.8)$
(b) $\mu = (0.3, 2)$
(c) $\mu = (0.5, 1)$
(d) $\mu = (0.6, 2.4)$

What is an emulator?
When can we emulate?\(^6\)

- \(\mathcal{M}_h\): manifold generated by simulator, contain solutions \(u_h(\mu^k)\)
- \(\mathcal{V}_N\): subspace generated by snapshots (design dataset) \(\{\xi_k\}\)
- \(u_N(\mu)\): approximated (emulated) solution.

What affects reducibility?
1. local vs. global dim.
2. smoothness of parameter dependence
3. approximability of \(\mathcal{M}_h\):
   Kolmogorov \(n\)-width \(\sim\) SV of snapshots

---

Kolmogorov $n$-width and SVD

Kolmogorov $n$-width: distance between the span of a subset of solution and the complete solution set.
Emulation in large models
What’s next?

Use the methods shown before for emulation of large models?

- Emulate the (low dimensional) relevant outputs
- Reduce the model locally

*reduced order model*: recovers all states (dof) ⊂ emulator
"Special" large domains
"Special" large domains
Divide and conquer

- Identify fundamental sub-domains
- Build sub-domain library
- Define coupling conditions(!)
- Simulate only in fundamental sub-domains (reduced model)
The master element

One domain to simulate them all and in the model bind them.
The master element

One domain to simulate them all and in the model bind them.
The master element

One domain to simulate them all and in the model bind them.
Dimensional heterogeneous models

• Some parts are not that important
• "Simplify" those
  • 1D phenomenological
  • 1D Emulator of detailed model
  • Reduced model
• Coupling is challenging
  • nD emulation on mesh nodes
Summary

- Methodological unification: GP, RR & KF → Fast emulator training on large datasets
- Division of large model domains in constituents
- Multi-scale modeling via parametrized emulators

Coupling is challenging!
Thank you!

Q&A