Estimating the Model Error in Dynamical Systems

Richard Wilkinson[1], Jeremy Oakley[2] and Tony O'Hagan[2]

[1]School of Mathematical Sciences, University of Nottingham
[2]Department of Probability and Statistics, University of Sheffield
r.d.wilkinson@nottingham.ac.uk

Exeter April 2010

Introduction

How can we estimate a model's error (discrepancy) from data?

- When using dynamical systems we are often in a sequential prediction setting where we make predictions before then observing the outcome.
- Embedded in this process is information about how well the model performs.
- We can specify a class of models for the error, and then try to learn about the error from our predictions and the realised data.

The work here is at an early stage and only been tried on simple models.

- Welcome any comments/questions etc throughout.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \ldots, x_T)$.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \dots, x_T)$.
- Computer model f which encapsulates our beliefs about the dynamics of the state vector

$$x_{t+1} = f_\phi(x_t, u_t)$$

  which depends on forcings u_t and parameters ϕ. We treat f as a black-box.

Mathematical Framework

Suppose we have

- State vector x_t which evolves through time. Let $x_{0:T}$ denote $(x_0, x_1, \dots, x_T)$.
- Computer model f which encapsulates our beliefs about the dynamics of the state vector

$$x_{t+1} = f_\phi(x_t, u_t)$$

  which depends on forcings u_t and parameters ϕ. We treat f as a black-box.
- Observations

$$z_t = h(x_t)$$

  where $h(\cdot)$ usually contains some stochastic element

In data assimilation we try to make use of the data and the computer model. If the model is wrong, we should take this into account.

Moving from white to coloured noise

Common data assimilation approach

- State evolution: $x_{t+1} = f_\phi(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.

Moving from white to coloured noise

Common data assimilation approach

- State evolution: $x_{t+1} = f_\phi(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.
- Aim is to find quantities such as
  - Filtering distribution $\pi(x_t|z_{0:t})$
  - Smoothing distribution $\pi(x_{0:T}|z_{0:T})$
  - Forward predictions $\pi(x_{T+t}|z_{0:T})$

  where $\pi(\cdot|\cdot)$ denotes a conditional probability distribution. We implicitly assume that these distributions are conditional on knowledge of forcings and the model, i.e., $\pi(x_t|z_{0:t}, f, \phi, u_{0:t})$

Moving from white to coloured noise

Common data assimilation approach

- State evolution: $x_{t+1} = f_\phi(x_t, u_t) + \epsilon_t$ where ϵ_t are iid rvs.
- Aim is to find quantities such as
  - Filtering distribution $\pi(x_t|z_{0:t})$
  - Smoothing distribution $\pi(x_{0:T}|z_{0:T})$
  - Forward predictions $\pi(x_{T+t}|z_{0:T})$

  where $\pi(\cdot|\cdot)$ denotes a conditional probability distribution. We implicitly assume that these distributions are conditional on knowledge of forcings and the model, i.e., $\pi(x_t|z_{0:t}, f, \phi, u_{0:t})$

Instead of the white noise model error considered above, we ask whether there is a stronger signal that could be learnt:

- State evolution: $x_{t+1} = f_\phi(x_t, u_t) + \delta(x_t, u_t) + \epsilon_t$
- Observations: $z_t = h(x_t)$ as above.

The hope is that by learning the model discrepancy we will produce more accurate predictions

- smaller error
- uncertainty bounds that we believe - fewer surprises.

The hope is that by learning the model discrepancy we will produce more accurate predictions

- smaller error
- uncertainty bounds that we believe - fewer surprises.

Why this is difficult?

- $x_{0:T}$ is usually unobserved, but given observations $z_{0:T}$ and a fully specified model we can infer $x_{0:T}$.
- When we want to learn the discrepancy $\delta(x)$ we are in the situation where we estimate δ from $x_{0:T}$, but where we must estimate $x_{0:T}$ from a description of δ.

Toy Example

Consider the following simple 1D example:

- Reality: $x_{t+1} = -x_t^2 + x_t + u_t$ with forcing model

$$u_t = \begin{cases} 0 \text{ w.p. } 0.6 \\ U \text{ w.p. } 0.4 \text{ where } U \sim Exp(10) \end{cases}$$

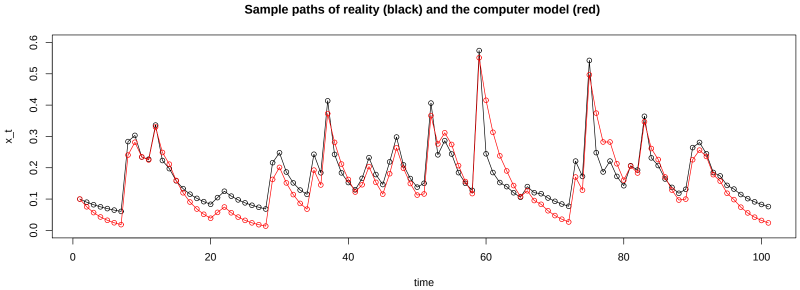
Toy Example

Consider the following simple 1D example:

- Reality: $x_{t+1} = -x_t^2 + x_t + u_t$ with forcing model

$$u_t = \begin{cases} 0 \text{ w.p. } 0.6 \\ U \text{ w.p. } 0.4 \text{ where } U \sim Exp(10) \end{cases}$$

- Computer model: $x_{t+1} = \phi x_t + u_t$



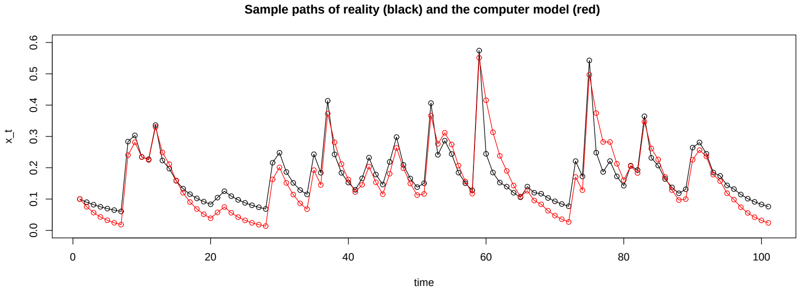
Toy Example

Consider the following simple 1D example:

- Reality: $x_{t+1} = -x_t^2 + x_t + u_t$ with forcing model

$$u_t = \begin{cases} 0 \text{ w.p. } 0.6 \\ U \text{ w.p. } 0.4 \text{ where } U \sim Exp(10) \end{cases}$$

- Computer model: $x_{t+1} = \phi x_t + u_t$
- Observations $z_t = x_t + V_t$ where $V_t \sim N(0, \sigma^2_{obs})$.



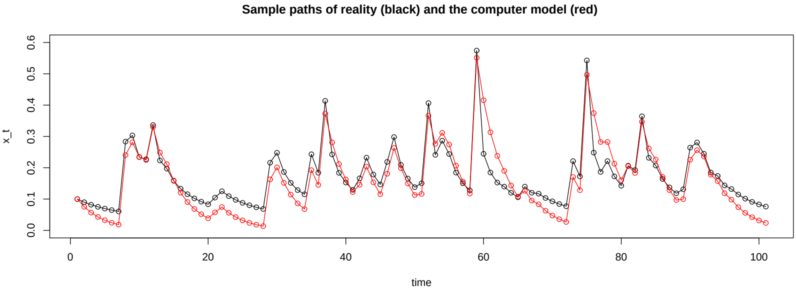
Toy Example

Consider the following simple 1D example:

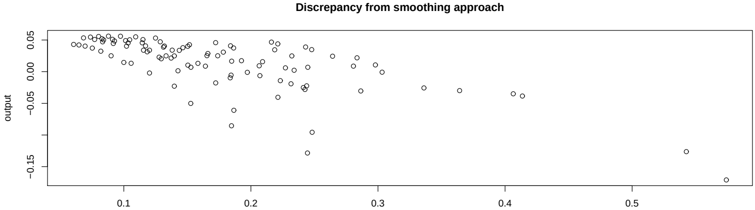
- Reality: $x_{t+1} = -x_t^2 + x_t + u_t$ with forcing model

$$u_t = \begin{cases} 0 \text{ w.p. } 0.6 \\ U \text{ w.p. } 0.4 \text{ where } U \sim Exp(10) \end{cases}$$

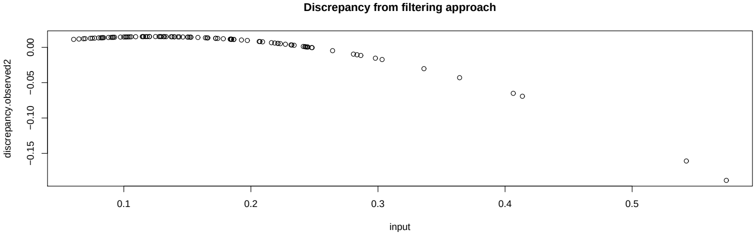
- Computer model: $x_{t+1} = \phi x_t + u_t$
- Observations $z_t = x_t + V_t$ where $V_t \sim N(0, \sigma^2_{obs})$.
- In this case the discrepancy is known exactly: $\delta(x) = x(1-\phi) - x^2$



The difficulty of learning δ is highlighted by looking at $x_{t+1} - f(x_t)$ where x_t is an uncorrected trajectory generated from the model.



At first sight this would suggest a white noise error isn't too bad.



Once we correct the state vector after each model time step a much clearer picture emerges.

Parametric approach

Start with a parametric model for δ, e.g., $\delta(x) = \sum_{i=0}^{p} \beta_i x^i + \epsilon$ where $\epsilon \sim N(0, \tau)$, with $\theta = (\tau, \beta_0, \ldots, \beta_p)$ unknown.

Parametric approach

Start with a parametric model for δ, e.g., $\delta(x) = \sum_{i=0}^{p} \beta_i x^i + \epsilon$ where $\epsilon \sim N(0, \tau)$, with $\theta = (\tau, \beta_0, \ldots, \beta_p)$ unknown.

- The problem now looks like a missing data problem:

$$\pi(x_{0:t}, z_{0:t}|\theta) = \pi(z_{0:t}|x_{0:t})\pi(x_{0:t}|\theta)$$

  is easy to work with when $x_{0:t}$ and $z_{0:t}$ are known. However $x_{0:t}$ is missing and $\pi(z_{0:t}|\theta)$ is unknown.

Parametric approach

Start with a parametric model for δ, e.g., $\delta(x) = \sum_{i=0}^{p} \beta_i x^i + \epsilon$ where $\epsilon \sim N(0, \tau)$, with $\theta = (\tau, \beta_0, \ldots, \beta_p)$ unknown.

- The problem now looks like a missing data problem:

$$\pi(x_{0:t}, z_{0:t}|\theta) = \pi(z_{0:t}|x_{0:t})\pi(x_{0:t}|\theta)$$

  is easy to work with when $x_{0:t}$ and $z_{0:t}$ are known. However $x_{0:t}$ is missing and $\pi(z_{0:t}|\theta)$ is unknown.
- The EM algorithm can be used to estimate the best fitting model for δ from the specified class of models.

An EM algorithm for estimating δ

We iterate between the E and M steps:

- E-step: Calculate

$$Q(\theta, \theta^{(m)}) = \mathbb{E}_{X_{0:T}} \left[\log \pi(X_{0:T}, z_{0:T} | \theta) \mid z_{0:T}, \theta^{(m)} \right]$$

- M-step: Maximize Q and set

$$\theta^{(m+1)} = \arg\max_\theta Q(\theta, \theta^{(m)})$$

An EM algorithm for estimating δ

We iterate between the E and M steps:

- E-step: Calculate

$$Q(\theta, \theta^{(m)}) = \mathbb{E}_{X_{0:T}}\left[\log \pi(X_{0:T}, z_{0:T}|\theta) \mid z_{0:T}, \theta^{(m)}\right]$$

  - This expectation is taken with respect to the distribution $\pi(x_{0:T} \mid z_{0:T}, \theta^{(m)})$

- M-step: Maximize Q and set

$$\theta^{(m+1)} = \arg\max_\theta Q(\theta, \theta^{(m)})$$

An EM algorithm for estimating δ

We iterate between the E and M steps:

- E-step: Calculate

$$Q(\theta, \theta^{(m)}) = \mathbb{E}_{X_{0:T}} \left[\log \pi(X_{0:T}, z_{0:T} | \theta) \mid z_{0:T}, \theta^{(m)} \right]$$

  - This expectation is taken with respect to the distribution $\pi(x_{0:T} \mid z_{0:T}, \theta^{(m)})$
  - This is the smoothing distribution from the fully specified model, and is not known analytically. However, it can be sampled from and the Monte Carlo expectation used for Q.

- M-step: Maximize Q and set

$$\theta^{(m+1)} = \arg\max_\theta Q(\theta, \theta^{(m)})$$

An EM algorithm for estimating δ

We iterate between the E and M steps:

- E-step: Calculate

$$Q(\theta, \theta^{(m)}) = \mathbb{E}_{X_{0:T}} \left[\log \pi(X_{0:T}, z_{0:T}|\theta) \mid z_{0:T}, \theta^{(m)} \right]$$

  - This expectation is taken with respect to the distribution $\pi(x_{0:T} \mid z_{0:T}, \theta^{(m)})$
  - This is the smoothing distribution from the fully specified model, and is not known analytically. However, it can be sampled from and the Monte Carlo expectation used for Q.

- M-step: Maximize Q and set

$$\theta^{(m+1)} = \text{arg max}_\theta Q(\theta, \theta^{(m)})$$

  - For the linear parametric model assumed here, it can be shown that this step reduces to fitting a linear regression model.

Comments

This gives a sequence $\theta^{(0)}, \theta^{(1)}, \ldots$ that tends to the maximum likelihood estimates $\text{argmax}_\theta \pi(z_{0:t}|\theta)$.

We can think of this as two steps which we loop around

1. Given an estimate for θ (and hence δ), estimate the true trajectory $x_{0:T}$ from $\pi(x_{0:T} \mid z_{0:T}, \theta)$.
2. Given samples from $\pi(x_{0:T} \mid z_{0:T}, \theta)$, estimate a value for θ.

The EM algorithm suggests that this converges to the mle (subject to problems with the expectation being approximated by a Monte Carlo sum).

Comments

This gives a sequence $\theta^{(0)}, \theta^{(1)}, \ldots$ that tends to the maximum likelihood estimates $\text{argmax}_\theta \pi(z_{0:t}|\theta)$.

We can think of this as two steps which we loop around

1. Given an estimate for θ (and hence δ), estimate the true trajectory $x_{0:T}$ from $\pi(x_{0:T} \mid z_{0:T}, \theta)$.
2. Given samples from $\pi(x_{0:T} \mid z_{0:T}, \theta)$, estimate a value for θ.

The EM algorithm suggests that this converges to the mle (subject to problems with the expectation being approximated by a Monte Carlo sum).

Restricting ourselves to a parametric family for δ is likely to be unsatisfactory in many cases. Is there a non-parametric version?

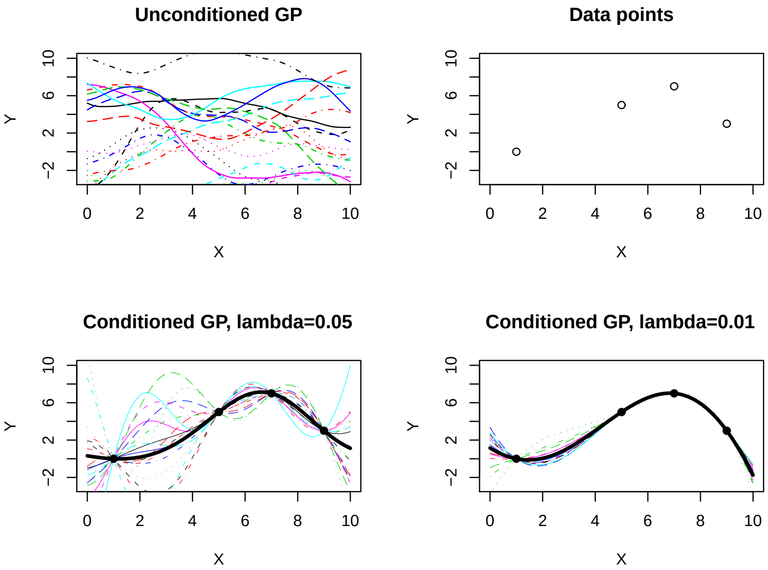
Gaussian Processes

Gaussian processes (GPs) provide a convenient semi-parametric model of functions.

- They are characterised by a prior mean and covariance functions, $m(x)$ and $c(x, x')$ respectively.
- Write $f \sim GP(m(\cdot), c(\cdot, \cdot))$ to denote that $(f(x_1), \ldots, f(x_n))^T$ has a multivariate normal distribution with mean $(m(x_1), \ldots, m(x_n))^T$ and covariance matrix with ijth entry $c(x_i, x_j)$.
- Importantly, the GP property is invariant under conditioning. That is, if $f \sim GP(m(\cdot), c(\cdot, \cdot))$ then conditional on observing the function value at various locations $\mathcal{D} = \{y_1 = f(x_1), \ldots, y_n = f(x_n)\}$ we find that f still has a Gaussian process distribution but with updated mean and covariance functions

$$f \mid \mathcal{D} \sim GP(m^*(\cdot), c^*(\cdot, \cdot))$$

Gaussian Processes



Iterative batch method for estimating δ

We can either use a Gibbs style sampler to estimate δ, or an EM type algorithm as before:

1. Given a model for the discrepancy, we can sample from $\pi(x_{0:T}|z_{0:T})$ using a particle smoother.

2. Given samples $x_{0:T}^{(1)}, \ldots, x_{0:T}^{(N)}$ we can fit a GP model to the observed discrepancies
$$d_i^j = x_{i+1}^{(j)} - f_\phi(x_i^{(j)}, u_i).$$

Iterative batch method for estimating δ

We can either use a Gibbs style sampler to estimate δ, or an EM type algorithm as before:

1. Given a model for the discrepancy, we can sample from $\pi(x_{0:T}|z_{0:T})$ using a particle smoother.
   For example, the method of Godsill *et al.* (2004) involves
   - A forward filtering sweep to find marginals $\pi(x_t|z_{0:t})$
   - Followed by a backward smoothing sweep to find smoothed posteriors $\pi(x_{0:T}|z_{0:T})$.
2. Given samples $x_{0:T}^{(1)}, \ldots, x_{0:T}^{(N)}$ we can fit a GP model to the observed discrepancies
$$d_i^j = x_{i+1}^{(j)} - f_\phi(x_i^{(j)}, u_i).$$

Filtering - $\pi(x_t|z_{0:t})$

The bootstrap filter

1. **Initialize** t=1
   For $i = 1, \dots, N$ sample $x_1^{(i)} \sim \pi(x_1)$, set $t = 2$
2. **Importance step**
   - For $i = 1, \dots, N$, sample
   $$\tilde{x}_t^{(i)} \sim \pi(x_t|x_{t-1}^{(i)}) \qquad \sim f(x_{t-1}) + \delta(x_{t+1})$$
   - Calculate the importance weights
   $$\tilde{w}^{(i)} \propto \pi(z_t|\tilde{x}_t^{(i)}) \qquad = \phi(z_t; x_t, \sigma^2_{obs})$$
3. **Selection step**
   - Sample with replacement N particles $(x_t^{(i)}, i = 1, \dots, N)$ from $(\tilde{x}_t^{(i)}, i = 1, \dots, N)$ according to the importance weights.
   - Set $t = t + 1$ and go to step 2. Reset all weights to be proportional to 1.

Smoothing $\pi(x_{0:T} \mid z_{0:T})$

Assume we have filtered particles $\{x_t^{(i)}\}_{i=1,\dots,N,t=1,\dots,T}$ with $x_t^{(i)} \sim \pi(x_t|z_{0:t})$ (assume all weights are $\propto 1$ because of gratuitous resampling in the filter).

Smoothing

- Choose $\tilde{x}_T = x_T^{(i)}$ at random from filtered particles at time T.
- For $t = T - 1$ to 1:
  - Calculate $w_{t|t+1}^{(i)} \propto \pi(\tilde{x}_{t+1}|x_t^{(i)})$ for each i
  - Choose $\tilde{x}_t = x_t^{(i)}$ with probability $w_{t|t+1}^{(i)}$

Then $\tilde{x}_{1:T}$ is an approximate realization from $\pi(x_{1:T}|z_{1:T})$.

Algorithm Summary

A heuristic algorithm for learning $\delta(\cdot)$

1. Using the white noise discrepancy model, draw sample trajectories $x_{0:T}^{(j)}$ from $\pi(x_{0:T}|z_{0:T})$.
2. Using these realizations, estimate values of $\delta_1(\cdot)$ and fit a Gaussian process model for δ_1.
3. At stage m, use discrepancy δ_m to sample from $\pi(x_{0:T}|z_{0:T}, \delta_m)$.
4. Use realizations $x_{0:T}^{(j)}$ from step 3 to estimate δ_{m+1}:

$$\delta_{m+1}(x_t^{(j)}) = x_{t+1}^{(j)} - f_\phi(x_t^{(j)})$$

5. Fit a GP model to these data. Return to step 3.

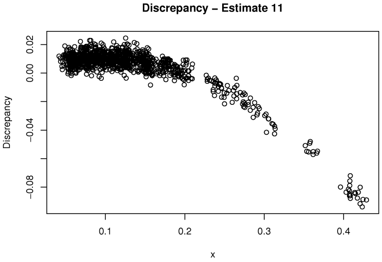
Fitting a GP discrepancy model

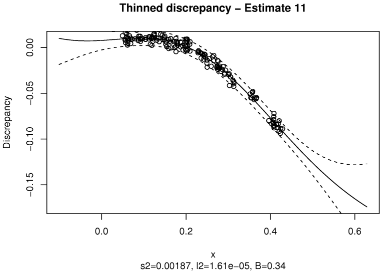
The particle filter and smoother require a reasonable number of particles in order to give accurate results. This leaves the problem of how to fit a GP to a large amount of noisy data.

Fitting a GP discrepancy model

The particle filter and smoother require a reasonable number of particles in order to give accurate results. This leaves the problem of how to fit a GP to a large amount of noisy data.

Thin by splitting the input range into intervals and sampling a small number of points from each interval.

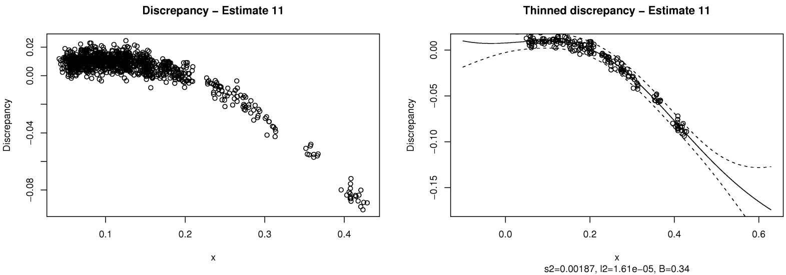




Fitting a GP discrepancy model

The particle filter and smoother require a reasonable number of particles in order to give accurate results. This leaves the problem of how to fit a GP to a large amount of noisy data.

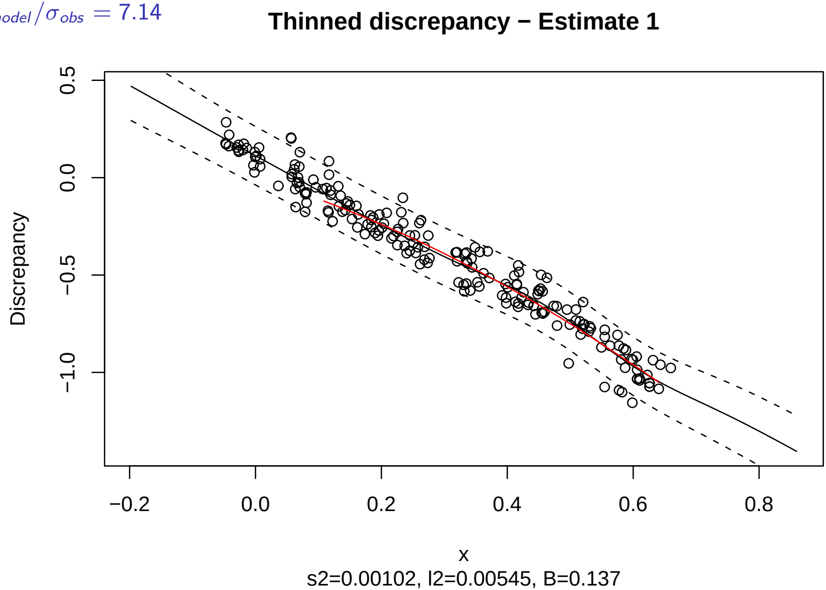
Thin by splitting the input range into intervals and sampling a small number of points from each interval.



Estimate length scale, variance and nugget term using max-likelihood or map estimates.

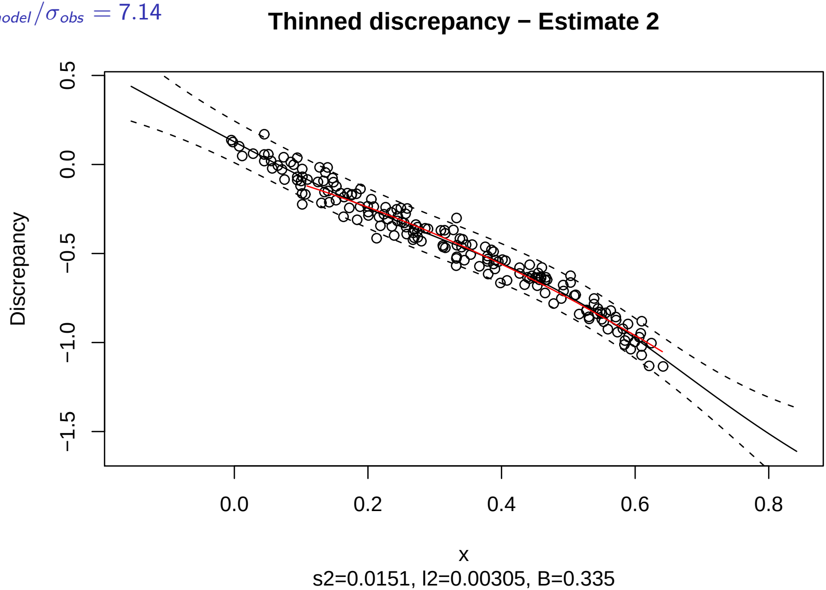
Results for toy model: sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



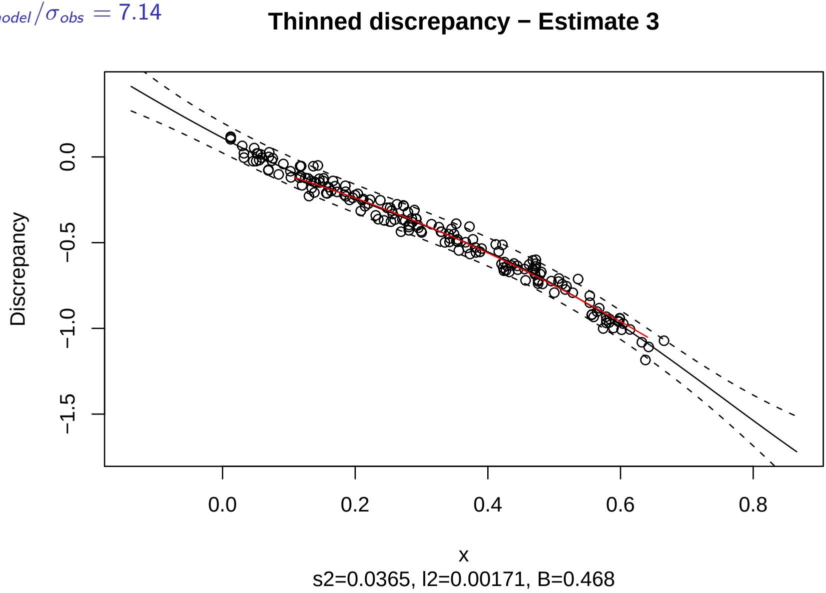
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



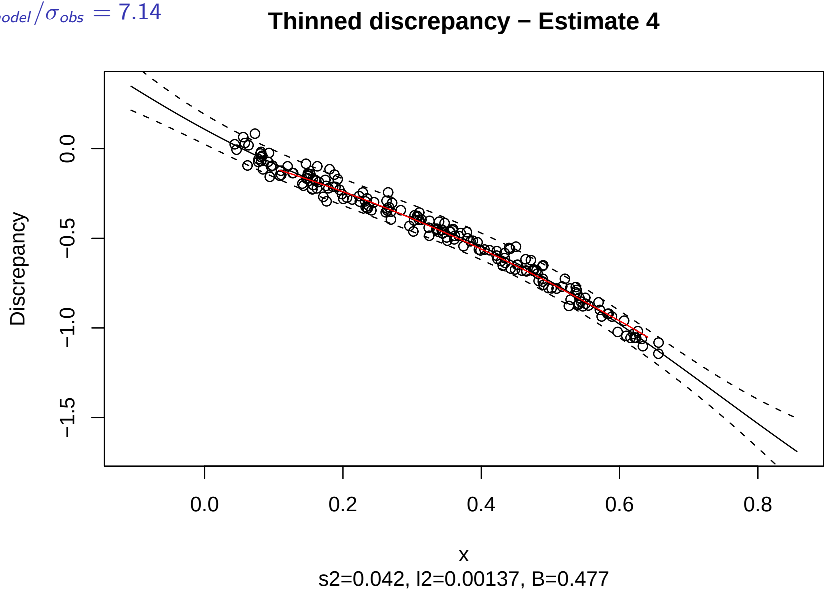
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



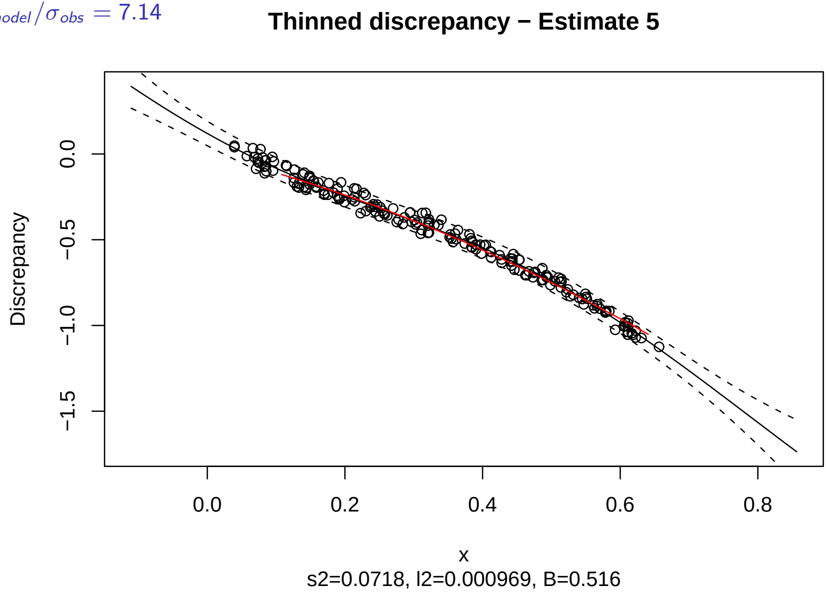
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



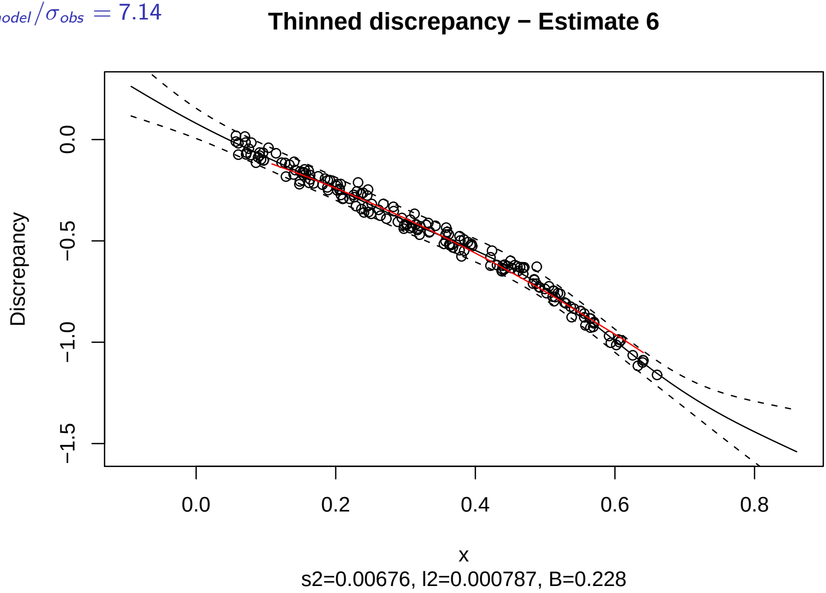
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



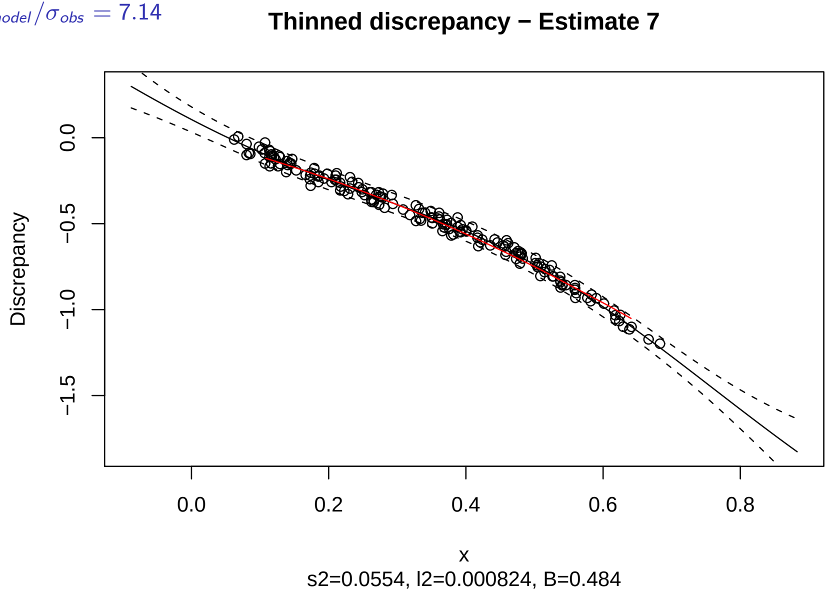
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



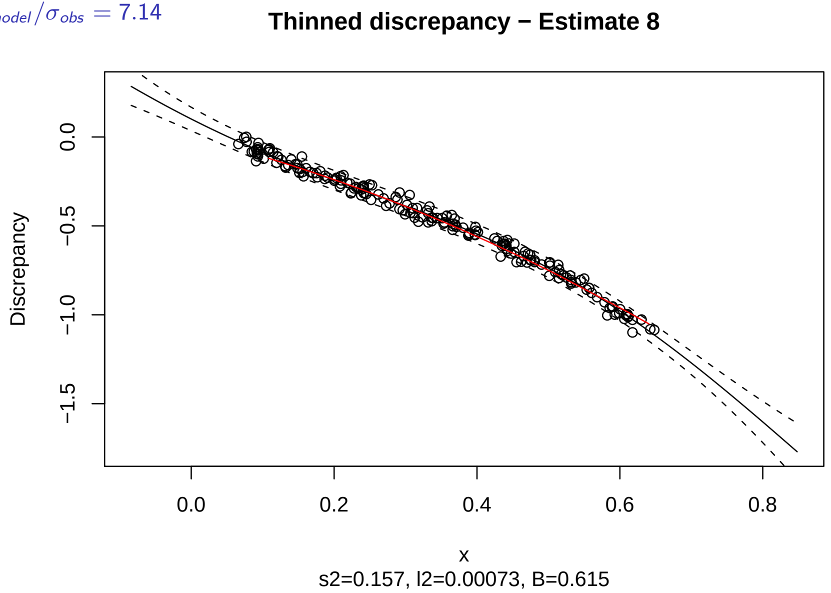
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



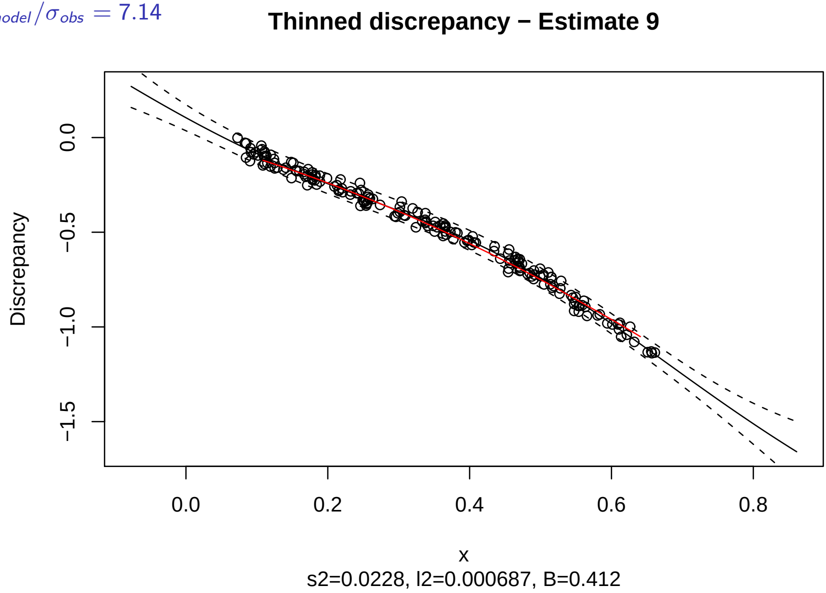
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



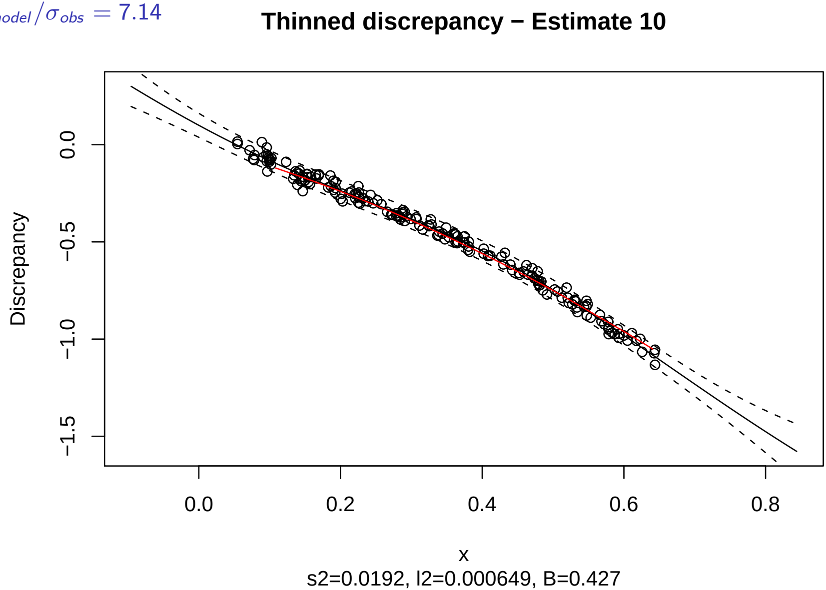
Sequence of discrepancy estimates

$\sigma_{model}/\sigma_{obs} = 7.14$



Sequence of discrepancy estimates

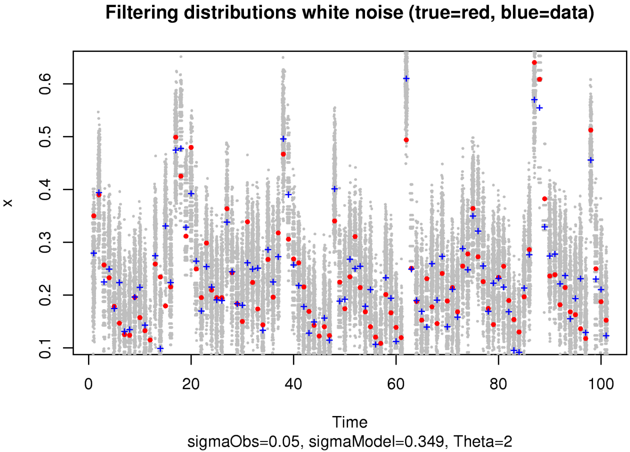
$\sigma_{model}/\sigma_{obs} = 7.14$



Filtering Distributions

$\sigma_{model}/\sigma_{obs} = 7.14$

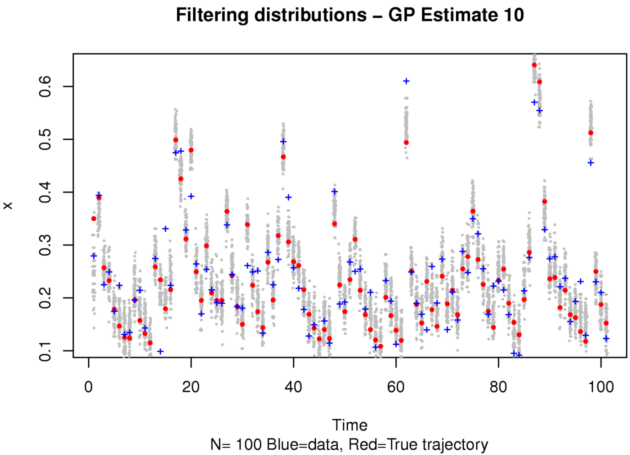
Move from



Filtering Distributions

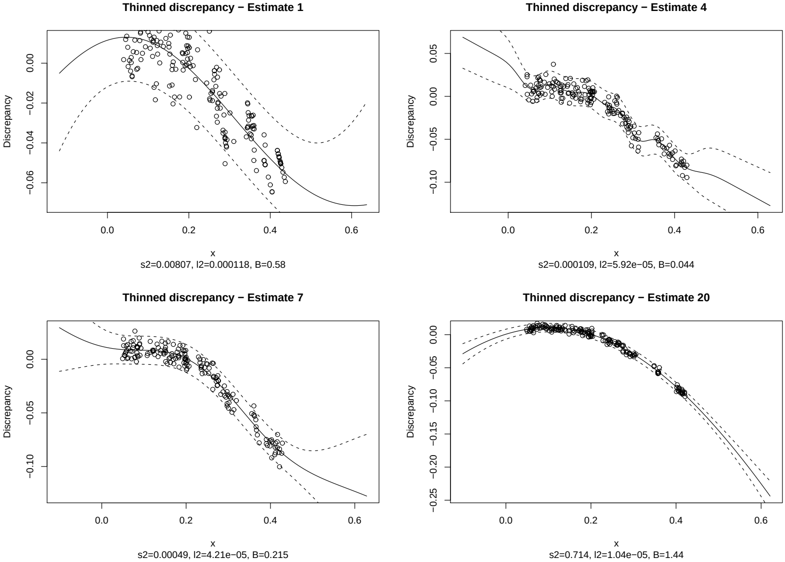
$\sigma_{model}/\sigma_{obs} = 7.14$

To



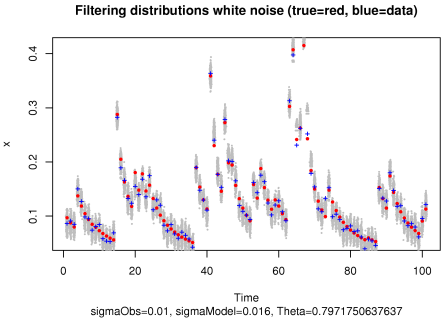
Similar sized model and measurement error

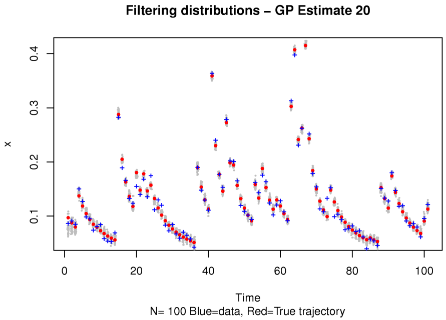
$\sigma_{model}/\sigma_{obs} = 1.6$



Similar sized (small) model and measurement error

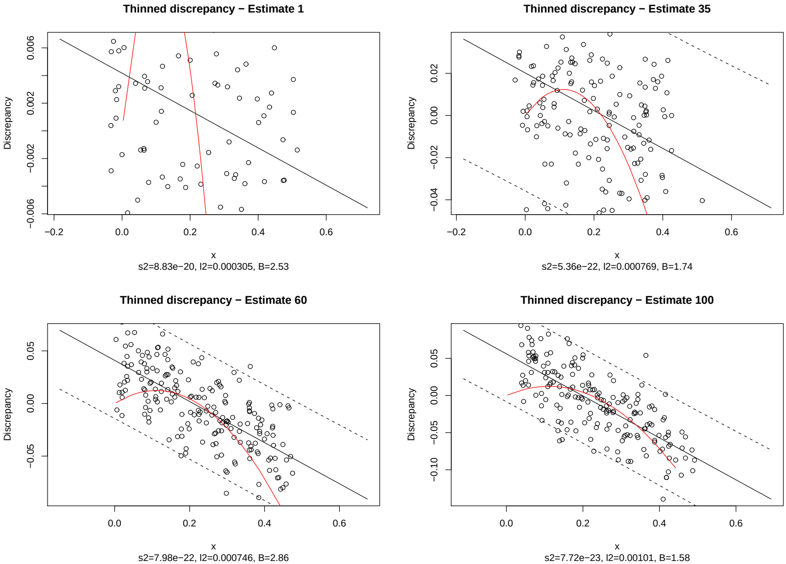
$\sigma_{model}/\sigma_{obs} = 1.6$





Small model error, large observation error

$\sigma_{model}/\sigma_{obs} = 0.036$



Concluding remarks

- Using a non-white model discrepancy can improve forecasts and state estimates. The discrepancy can be learnt from observation alone.
- Approach is computationally intensive. Even for these toy models, 100 iterations of the algorithm can take a few minutes.
- In higher dimensional problems (esp. if $\dim(x) > \dim(z)$) it can be hard to discover if a systematic model error exists.
- Our aim is to perform Bayesian inference (data are conditioned on; use all available information), but we compromise in places:
  - Coherent subjectivists are well-calibrated in the long-run (Dawid 1984). However, when we use complex models, it is hard/impossible to be a good Bayesian - we tend to operate in periods of revolution. We tend to use a model for a while until it is discarded or upgraded and then stick with the new version for a while - calibration scores can be useful tools.
  - Sequential approaches are extremely costly, which is why we've used a batch approach here.
  - If the modellers have beliefs about the shape of the model error, it is possible to incorporate this into our *a priori* description of the GP model.

Thank you for listening!