ODE parameter estimation

Matthieu Poyer

26 juillet 2019

Content

Content

Introduction : presentation of our problem

We have a model which is governed by this equation :

$$\boxed{d\mathbf{x}_t = f(t, \mathbf{x}_t, \theta)dt},$$

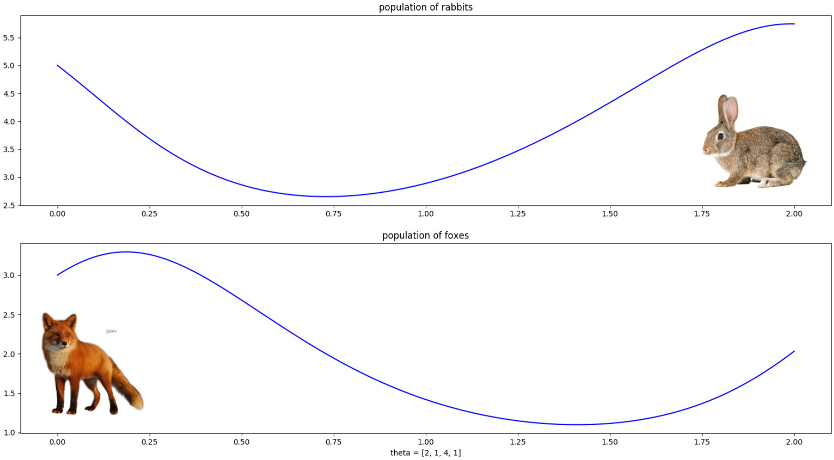
and the question is **how to find the parameter θ using the observations ?**

Introduction add noise in the equation

$$\boxed{d\mathbf{x}_t = f(t, \mathbf{x}_t, \theta)dt + \sigma dB_t}$$

- add some noise : σdB_t
- The unknown is θ and maybe σ too.

An example : the Lotka Volterra model



An example : the Lotka Volterra model

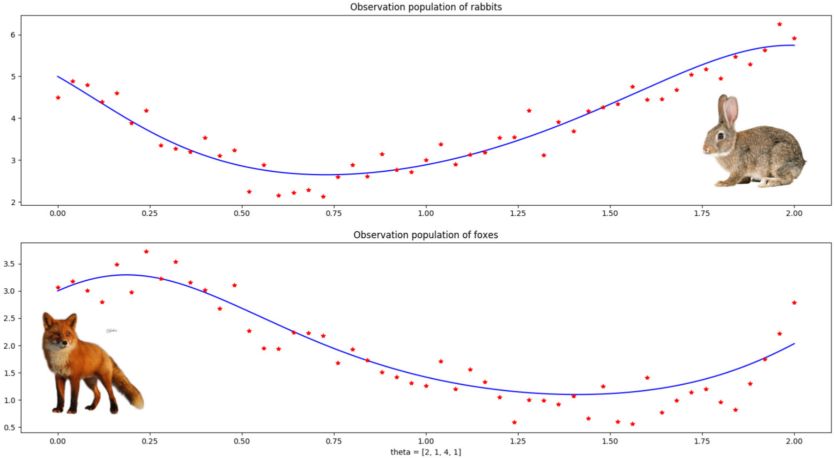
$$\boxed{d\mathbf{x}_t = f(t, \mathbf{x}_t, \theta)dt}$$

The problem that I used to test the theory is the Lotka-Volterra model :

$$\boxed{\begin{cases} \frac{dX_R}{dt} = X_R(\alpha - \beta X_F) \\ \frac{dX_F}{dt} = -X_F(\delta - \gamma X_R). \end{cases}}$$

So here the unknow are $\theta = (\alpha, \beta, \gamma, \delta)$, $\mathbf{x} = (x_R, x_F)$ and f does not depend on t.

An example : the Lotka Volterra model



Content

Likelihood of **Y** (observations) for given parameters

Likelihood :

$$L(\theta) = p(\mathbf{y}_{t_0}, t_0; \theta) \prod_{i=1}^{n} p(\mathbf{y}_{t_i}, t_i \mid \mathbf{y}_{t_{i-1}}, t_{i-1}; \theta)$$

where p is the probability to obtain $\mathbf{y}_{t_i}$ at t_i knowing $\mathbf{y}_{t_{i-1}}$ when f is paramtrisezd by θ. So its log-likelihood becomes :

$$\boxed{l(\theta) = \log(p(\mathbf{y}_{t_0}, t_0; \theta)) + \sum_{i=1}^{n} \log(p(\mathbf{y}_{t_i}, t_i \mid \mathbf{y}_{t_{i-1}}, t_{i-1}; \theta))}$$

$$\widehat{\theta}_n = \underset{\theta \in \Theta}{\arg\max}\ L(\theta).$$

Assumptions and Approximations

We do 2 hypothesis

- $\exists \Delta t \; \forall i, \;\; t_i - t_{i-1} = \Delta t$
- Euler approximation

$$\mathbf{x}_{t_{i+1}} = \underbrace{\mathbf{x}_{t_i}}_{\text{older value}} + \underbrace{f(t_i, \mathbf{x}_{t_i}, \theta)}_{\text{gradient}} \underbrace{\Delta t}_{\text{time step}} + \underbrace{\sigma(B_{t_{i+1}} - B_{t_i})}_{\text{noise}} .$$

The log-likelihood becomes :

$$l(\theta) = -\frac{1}{2} \sum_{i=1}^{n} \left(\frac{\|\sigma^{-1}(\mathbf{y}_{t_i} - \mathbf{y}_{t_{i-1}} - f(t_{i-1}, \mathbf{y}_{t_{i-1}}, \theta)(\Delta t))\|_2^2}{\Delta t} + \text{cste} \right) .$$

The new problem

So the new problem is how to minimise :

$$\boxed{S(\theta) = \sum_{i=1}^{n} \|\sigma^{-1}(\mathbf{y}_{t_i} - \mathbf{y}_{t_{i-1}} - f(t_{i-1}, \mathbf{y}_{t_{i-1}}, \theta)(\Delta t))\|_2^2.}$$

and this is much more easy.

Remark : So, in our case, the max-likelihood estimator is the least squared estimator.

The Gradient descent method

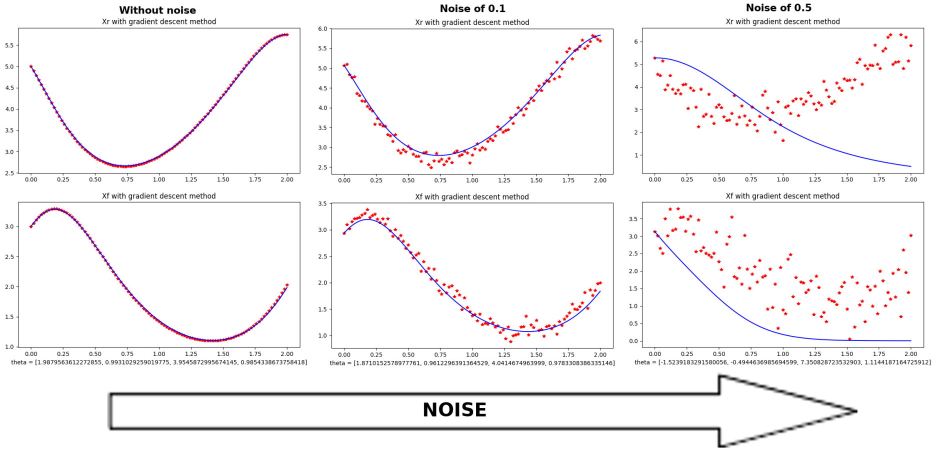
This algorithm uses this formula :

$$\boxed{\theta^{(\text{new})} = \theta - \alpha' \ \nabla_\theta S}$$

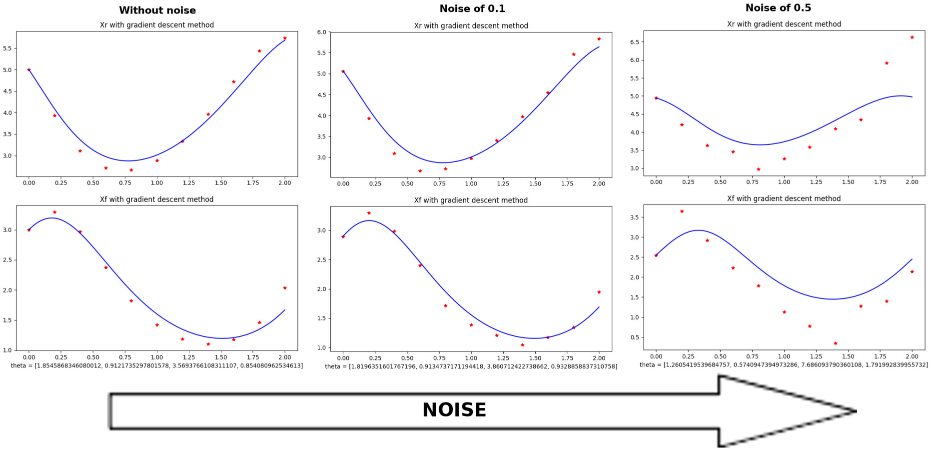
where $\nabla_\theta S$ is the gradient of S at θ and α' need to be determined.
We choose α' such that :

$$\alpha' = \arg\min_{\alpha \in \mathbb{R}} \{S(\theta - \alpha \nabla_\theta S)\}$$

The Lotka-Volterra model with many observations



The Lotka-Volterra model with few observations



The Gauß-Newton method

- The Newton method

$$\boxed{\theta^{(\text{new})} = \theta - \text{Hess}_\theta(S)^{-1}\nabla_\theta(S)}$$

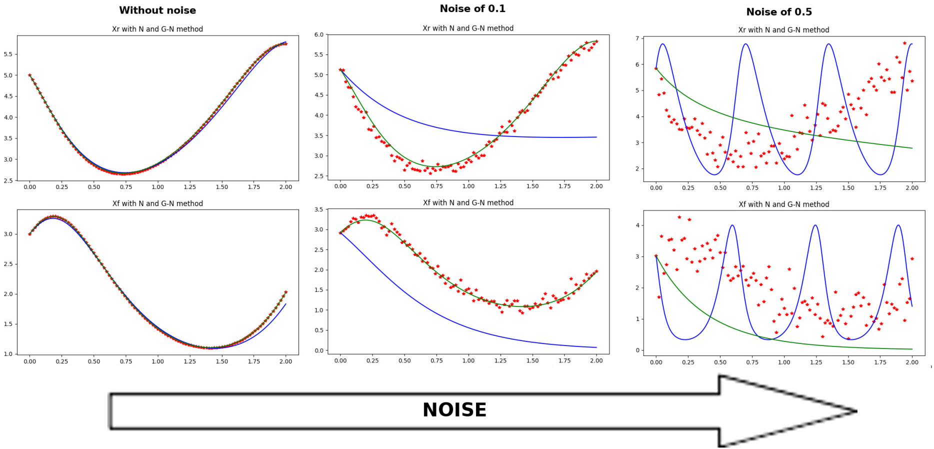
- The Gauß-Newton method :

$$\boxed{\theta^{(\text{new})} = \theta - (\text{Jac}_\theta(\mathbf{r})^\top \text{Jac}_\theta(\mathbf{r}))^{-1}\text{Jac}_\theta(\mathbf{r})^\top \mathbf{r}},$$

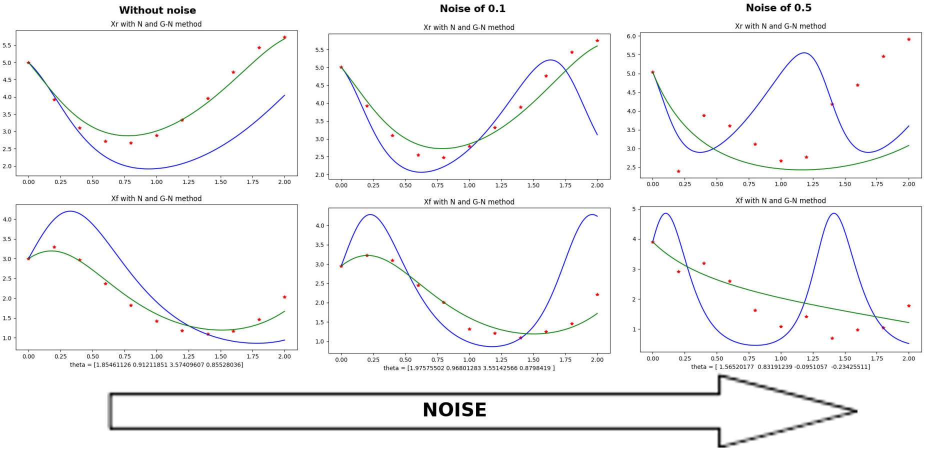
where $r_k = \|\sigma^{-1}(x_{t_k} - x_{t_{k-1}} - f(t_{k-1}, x_{t_{k-1}}, \theta)(\Delta t))\|_2$ and :

$$\mathbf{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix}.$$

The Lotka-Volterra model with few observations



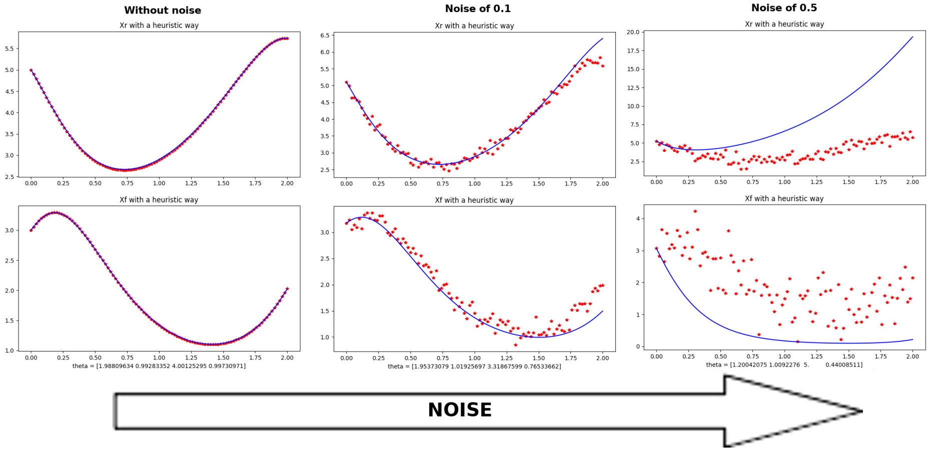
The Lotka-Volterra model with few observations



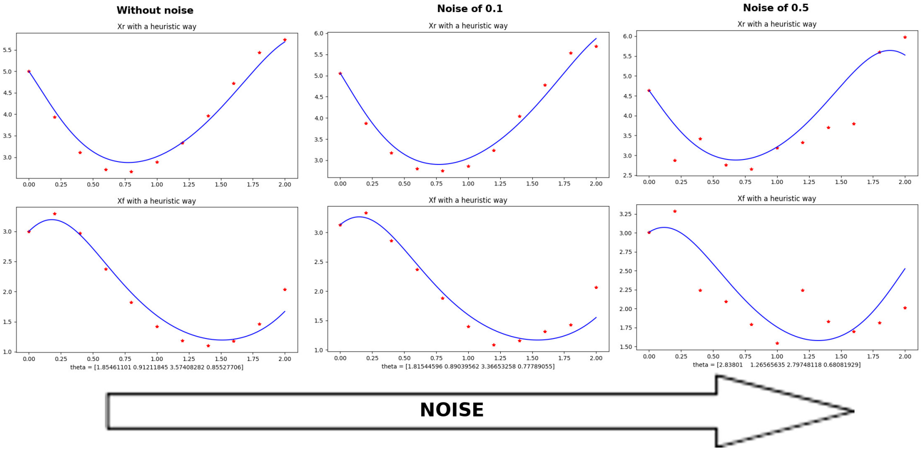
A heuristic method

We also can use **scipy.optimize.differential_evolution()**, this function permits to find a global minimum of a multivariate function. In our case it is S

The Lotka-Volterra model with few observations



The Lotka-Volterra model with few observations



Content

Bayesian approach

We also can use a Bayesian approach to answer the problem.

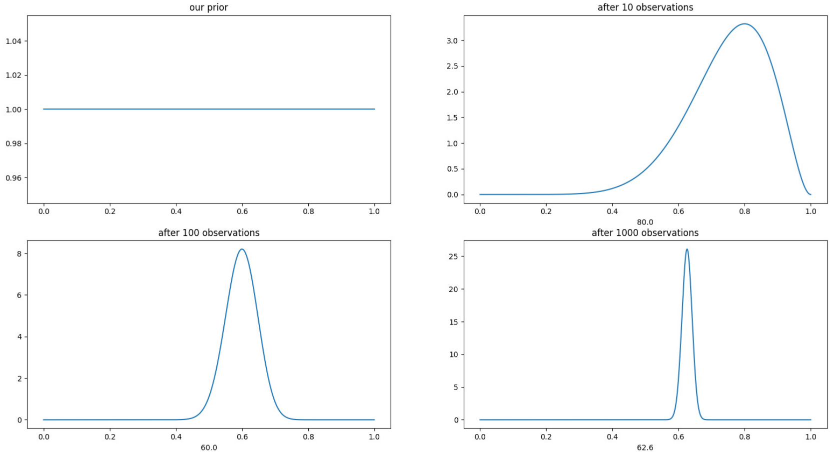
What is a bayesian approach ?

A bayesian approach consists on giving a law (prior) and adjust this law with the observation.

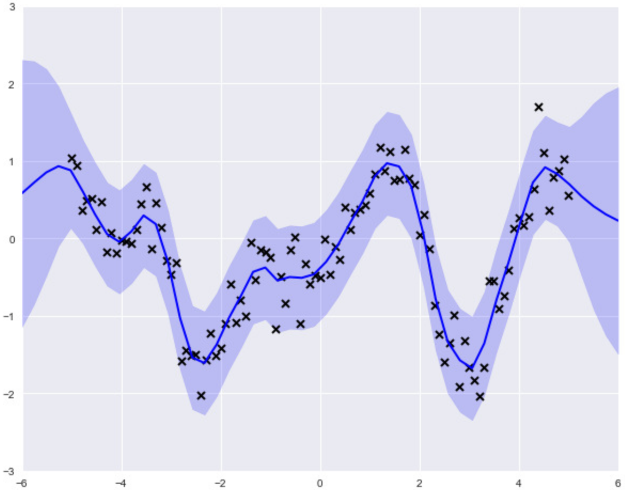
$$\boxed{\mathrm{p}(\mathrm{M} \mid \mathrm{D}) = \frac{\mathrm{p}(\mathrm{D} \mid \mathrm{M})\ \mathrm{p}(\mathrm{M})}{\mathrm{p}(\mathrm{D})}}$$

Bayesian approach

Example :

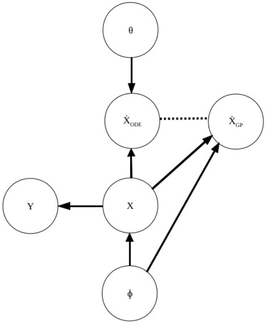


Gaussian Process



Gradient Matching

The main idea is to use a Gaussian Processes (GP) as a prior :



where θ are
the parameters that we are searching,
ϕ is our prior : the GP.

$$p(\phi, \mathbf{x}, \dot{\mathbf{x}}_{ODE}, \dot{\mathbf{x}}_{GP}, \mathbf{y}, \theta \mid \dot{\mathbf{x}}_{ODE} = \dot{\mathbf{x}}_{GP}) = p(\phi)p(\mathbf{x} \mid \phi)p(\dot{\mathbf{x}}_{GP} \mid \phi, \mathbf{x})p(\theta) p(\dot{\mathbf{x}}_{ODE} \mid \mathbf{x}, \theta)p(\mathbf{y} \mid \mathbf{x})\delta(\dot{\mathbf{x}}_{ODE} - \dot{\mathbf{x}}_{GP})$$

Gradient Matching

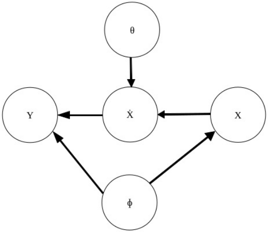
$$p(\phi, \mathbf{x}, \dot{\mathbf{x}}, \mathbf{y}, \theta) = \\ p(\phi)p(\mathbf{x} \mid \phi)p(\dot{\mathbf{x}}_{GP} \mid \phi, \mathbf{x})p(\theta)p(\dot{\mathbf{x}}_{ODE} \mid \mathbf{x}, \theta)p(\mathbf{y} \mid \mathbf{x})\delta(\dot{\mathbf{x}}_{ODE} - \dot{\mathbf{x}}_{GP})$$

Now we change our priors thanks :

$$\phi \sim p(\phi \mid \mathbf{y})$$
$$\mathbf{x} \sim p(\mathbf{x} \mid \phi, \mathbf{y})$$
$$\theta \sim p(\theta \mid \mathbf{x}, \phi)$$

An othe bayesian approach

An other approach is :

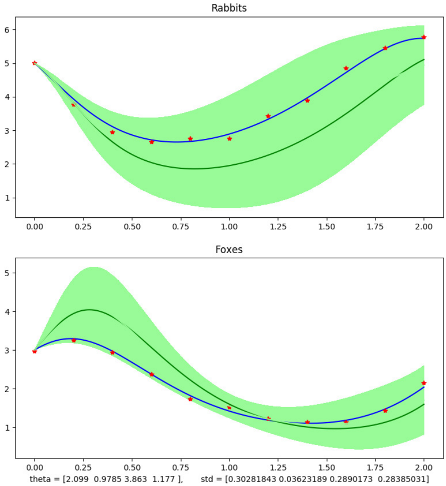


where θ are
the parameters that we are searching,
ϕ is our prior : the GP.

$$p(\mathbf{y}, \mathbf{x}, \dot{\mathbf{x}}, \phi, \theta) =$$
$$p(\phi)p(\mathbf{x} \mid \phi)p(\dot{\mathbf{x}} \mid \theta, \mathbf{x})p(\theta)p(\mathbf{y} \mid \dot{\mathbf{x}}, \phi)$$

$$\phi, \theta \sim p(\phi, \theta \mid \mathbf{y}, \mathbf{x})$$
$$\mathbf{x} \sim p(\mathbf{x} \mid \phi, \theta, \mathbf{y})$$

The Lotka-Volterra model with noise (0,1)



Content

Conclusion

methods	pro	contra
Gradient descent method	often fast	gradient big noise sensitive
Gauß-Newton method	fast (O(data)) often better than gradient descent	Jacobian noise sensitive
Newton method	fast (O(data)) better than G-N	Hessian big noise sensitive
Heuristic method	fast	heuristic
Bayesian approach	guess the noise	slow (O(data ^{3}))

Content

You can find this presentation and my python codes here :
`https://github.com/Yellowkholle/ODE-parameter-estimation-python`

Bibliography

- *Gaussian Processes for Bayesian Estimation in Ordinary Differential Equations* (Barber, Wang)
- *ODE parameter inference using adaptive gradient matching with Gaussian processes* (Dondelinger, Husmeier, Rogers, Filippone)
- *Accelerating Bayesian Inference over Nonlinear Differential Equations with Gaussian Processes* (Calderhead, Girolami, Lawrence)