# Notes: melting pot about machine learning

Robin Delabays

May 8, 2019

# 1 ML algorithms

## 1.1 Kernel methods

See Florian's notes [1] and the survey [2].

## 1.2 Reservoir computing

Good summary in [3].

# 2 Observations

Our aim is to predict the behavior of a (partially) unknown dynamical system, using algorithms of machine learning. We want to compare the performances of the algorithms described in Sec. 1. Further objective would be to evaluate how long the algorithm manage to mimic the dynamical system once it does not have real-time measurement anymore.

## 2.1 LTI system

We consider the Linear Time Invariant (LTI) system, subject to Gaussian noise, with unkown initial conditions,

$$\dot{x} = Ax\,, \qquad\qquad y = Cx + \epsilon\,, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$. If the initial conditions are $x_0$, then

$$y(t) = C\exp(At)x_0 + \epsilon(t)\,. \tag{2}$$

We will use

$$A = \begin{pmatrix} 0 & 1 \\ -0.1 & 0 \end{pmatrix}\,, \qquad C = \begin{pmatrix} 0 & 1 \end{pmatrix}\,, \qquad \epsilon \sim \mathcal{N}(0, 0.01)\,. \tag{3}$$

As shown by Florian, kernel methods work well for such systems, provided we know $A$ and $C$, and using pairs $\{(t_i, y(t_i)\}_{i=1,...,T}$ as training data. Actually, we can verify that this works quite well already for rather small $T \in \{5, ..., 10\}$.

Our reservoir computer works very poorly on this specific problem (see Fig. 1), even without noise. The training times were taken randomly (not at fixed intervals).

Our reservoir computer works much better if we gives training pairs $\{(y(t_0+k\Delta t), y(t_0+(k+\Delta k)\Delta t))\}_{k=1,...,T}$, i.e., system measurement with a time delay of $\Delta k\Delta t$. In Fig. 2, we gave the training outputs as inputs for the systems prediction, but without noise. When noise is added, this does not work very well anymore. Fig. 3 gives the same estimation with noise $\epsilon \sim \mathcal{N}(0, 0.01)$ with $\Delta k = 1$.

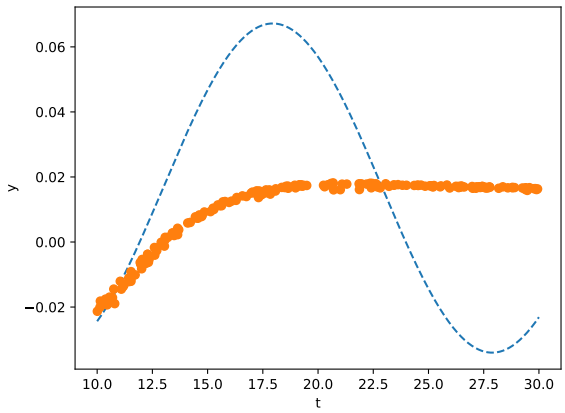


Figure 1: Trajectory of $y$ for the LTI system Eq. (3) (blue dashed line) and the reservoir computing estimation of it (orange dots). Inputs are random times. $N = 1000$, $D = 20$, $\rho = 1$, $\sigma = 0.01$, $\alpha = 1$, $\xi = 1$, $\beta = 1$.

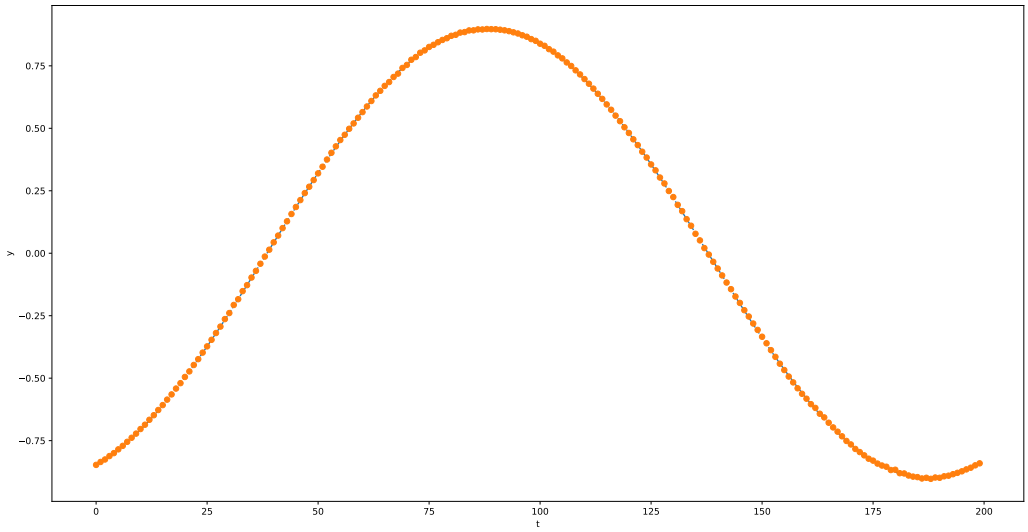


Figure 2: Trajectory of $y$ for the LTI system Eq. (3) (blue dashed line) and the reservoir computing estimation of it (orange dots). Inputs are previous system's state. $N = 1000$, $D = 20$, $\rho = 1$, $\sigma = 0.01$, $\alpha = 1$, $\xi = 1$, $\beta = 1$, $\Delta k = T_{\text{simu}}$.

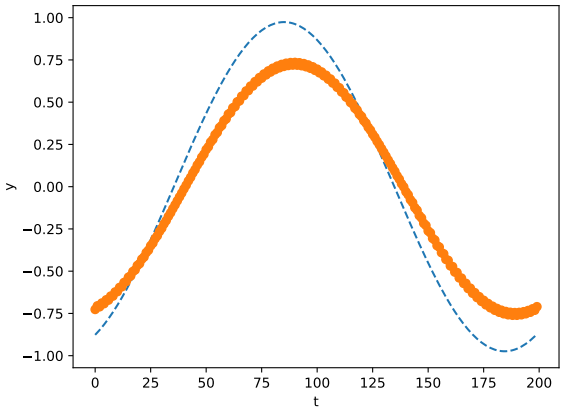


Figure 3: Same as Fig. 2 with noise. $N = 1000$, $D = 20$, $\rho = 1$, $\sigma = 0.01$, $\alpha = 1$, $\xi = 1$, $\beta = 1$, $\Delta k = 1$.

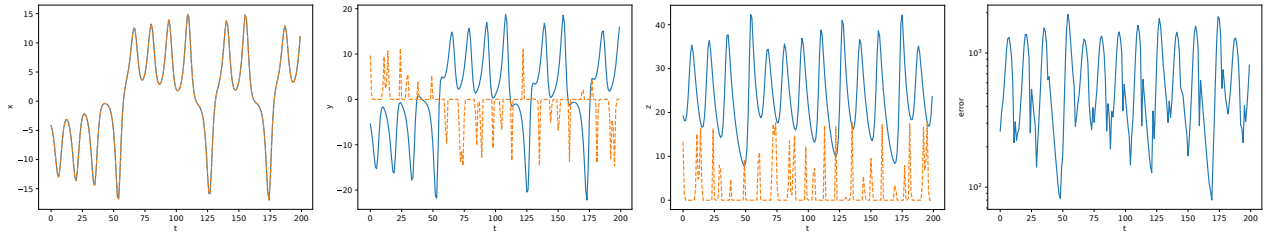

Figure 4: Trajectory of the Lorentz system Eq. (4) (blue line) and its "prediction" by Gaussian kernel method (orange dashed line). The variable $x$ is exact because it is the input. $T = 5000$, $\rho = 1$, $\gamma = 10^{-9}$.

### 2.2 Lorentz system

According to [3], the Lorentz system

$$\dot{x} = \sigma(y - x)\,, \qquad \sigma = 10\,, \tag{4}$$
$$\dot{y} = x(\rho - z) - y\,, \qquad \rho = 28\,, \tag{5}$$
$$\dot{z} = xy - \beta z\,, \qquad \beta = 8/3\,, \tag{6}$$

is well predicted by reservoir computing, using the measurement of $x$ as input and the other two variables $y$ and $z$ as output. We verified this.

Until now I have not been able to find a satisfying kernel to predict it using kernel methods. Kernels based on a basis of monomials (up to degree 4) as well a Gaussian kernels work poorly (Fig. 4).

## References

[1] Florian. *Notes on regularized regression, Gaussian processes, and Kalman Filtering* .

[2] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung. *Automatica* **50**, 657 (2014).

[3] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott. *Chaos* **27**, 041102 (2017).