Recursive Estimation of Global Models for Embedding Surfaces

Eric Métois
Information and Entertainment Group, Media Laboratory, Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139.

Motivated by the analysis and representation of musical sounds through non-linear models, we examine the use of a Kalman filter as an estimator of a polynomial prediction function for time series analysis. The work that we present in this paper is not specific to the nature of our data and the method that we suggest can be applied to any time series as long as it is the result of a fairly low dimensional dynamical system. We review the mechanism of a standard Kalman filter and evaluate the behavior of the estimated predictors.

1. Introduction

1.1. State space and lag space

Let's consider an autonomous dynamical system described by its state variables $\underline{\mathbf{x}}$ related to each other in a general fashion:

$$\frac{d\underline{\mathbf{x}}}{dt} = f(\underline{\mathbf{x}}).$$

The evolution of the system from a given initial state can be monitored by the trajectory of the vector $\underline{\mathbf{x}}$ as time passes by. This vector $\underline{\mathbf{x}}$ lives in the **state space** and the observation of this trajectory can teach us a lot about the internal mechanism of this dynamic system (i.e. about the relationships f). However, the nature and even the number of these internal states (or degrees of freedom) are usually unknown and we only have access to a subset of them if not only one. Let's suppose the only observation we have is a single variable z. Even though the dimension of our observation is one, we can choose to build a vector of arbitrary dimension d by using lag values of z: $\underline{\mathbf{y}}(t) = (z(t), z(t-\tau), \ldots, z(t-(d-1)\tau))^T$. This vector lives in the **lag space** in which it will draw another trajectory as time passes by.

1.2. The embedding theorem

The embedding theorem, proved by Floris Takens in 1981 in his paper "Detecting strange attractors in turbulence" [1] states that the state and the lag spaces are generically diffeomorphic given that the dimension d of the lag space is large enough. The trajectory of a 1D observation in a d-dimensional lag space and the trajectory of the entire system in its state space differ only by a smooth local change of coordinates. This result being a "generic" property means that it is not always true but that the set of cases for which it'll break is of probability 0. In other words, perturbing an unlucky case in an infinitely small way will make the result hold. Representing the data in lag space is not the unique way to obtain an embedding but it is very common because of its convenience.

In the context of modeling, classification or resynthesis, this result tells us that there is no need for "hidden variables" other than the lag values of the time series we are studying. Furthermore, the number of necessary lag values is directly related to the number of the system's degrees of freedom and this number can also be estimated by measuring statistics on the initial observation.

The state space of a system is an object we will never have access to whereas the lag space doesn't give us any such trouble. This strong relationship between these two objects allows us to forget the state space from now on and to manipulate lag values of an observation as if they were directly state variables of our system.

1.3. Dimensions

The sufficient dimension d of the lag space and the number of degrees of freedom are now the same thing. Let's consider a deterministic system that produces the discrete time observation $z_n=z(n\tau)$. By "deterministic", we mean here that the past values of this observation allow the prediction of its future values with no error. This can be written in terms of probability distributions as:

$$p(z_n | z_{n-1}, z_{n-2}, \ldots) = \delta(z_n - f(z_{n-1}, z_{n-2}, \ldots))$$

For this system to have a finite number of degrees of freedom, there has to be a dimension d such that:

$$p(z_n | z_{n-1}, z_{n-2}, \ldots) = p(z_n | z_{n-1}, z_{n-2}, .., z_{n-d}) = \delta(z_n - f_d(z_{n-1}, z_{n-2}, .., z_{n-d}))$$

(One can comprehend this property as a generalization of Markovianity.)

If such a dimension exists, and if our observation is a sampled strict sense stationary stochastic processes, then we'd have:

$$\begin{aligned} p_{k+1}(\tau) &= p(z_n, z_{n-1}, z_{n-2}, .., z_{n-k}) \\ &= \delta(z_n - f_d(z_{n-1}, z_{n-2}, .., z_{n-d})).p(z_{n-1}, z_{n-2}, .., z_{n-k}) \\ &= \delta(z_n - f_d(z_{n-1}, z_{n-2}, .., z_{n-d})).p_k(\tau) \qquad \text{for any } k \geq d \end{aligned}$$

Therefore if $H_k(\tau)$, $I_k(\tau)$ and $R_k(\tau)$ are respectively the entropy, the mutual information and the redundancy of k successive samples z_n of the observation with τ as the sampling period, for any $k \geq d$, we'd have:

$$\begin{aligned} H_{k+1}(\tau) &= H_k(\tau) \\ I_{k+1}(\tau) &= (k+1)H_1(\tau) - H_{k+1}(\tau) = H_1(\tau) + I_k(\tau) \\ R_{k+1}(\tau) &= I_{k+1}(\tau) - I_k(\tau) = H_1(\tau) \end{aligned}$$

In that case, this dimension is referred to as the **embedding dimension**. A natural way to determine this embedding dimension is therefore by evaluating the observation's joint entropy for various successive dimensions and by watching its evolution as the dimension increases. The estimation of these entropies can be a difficult and tricky task given the definition of entropy and its lack of continuity from discrete to continuous types random processes. The estimated values of these entropies can be very different whether we use our sampled, fixed resolution, data to create histograms or to estimate a parametric form of a continuous type probability mass function.

2. Local versus global models

From now on, we will assume the existence of an embedding dimension. We know that the existence of such an object might not appear from our data as clearly as we'd like. This dimension d will be the result of a fairly arbitrary decision rather than an unquestionable observation. This assumption can be stated as:

$$p(z_n | z_{n-1}, z_{n-2}, .., z_{n-d}) = \delta(z_n - f(z_{n-1}, z_{n-2}, .., z_{n-d}))$$
$$\text{or simply } z_n = f(z_{n-1}, z_{n-2}, .., z_{n-d})$$

Our system is therefore entirely characterized by a set of d initial conditions and a representation of the function f(). In order for our model to generalize the behavior of our system with variations on the initial conditions for instance, the representation of f() should be defined on a wider set than the training data (i.e. our observation). Given also that we want to avoid a prohibitive size of this representation, the goal of this training should be to estimate a parametric form for f().

For this purpose, there are two basic approaches, the global and the local approaches. A global approach assumes some fixed architecture for a closed form of the function f() on the entire set on which it's defined, and tunes the parameters of this architecture in order to fit the training data by minimizing some criteria. An example would be to fit a polynomial of dimension d and fixed order N to the observation with a least mean square criteria. A local approach will typically use the training data as the model and an interpolating method as the mean to generalize it. Local linear modeling is an example of a local approach. These two sets of approaches are not purely exclusive as one can choose to take a local approach on a representative subset of the training data. Each one of these representative points carries some information about the system's behavior in the corresponding neighborhood to the rest of the model. These points are sometimes referred to as the anchor points of radial basis functions. If the number of anchor points is fixed, than there is an assumption concerning the closed form of the model even though it is based on interpolations between observed data.

A local approach usually gives better performances as it doesn't constraint the model as much as a global method would. Yet, the representation it provides is almost as big as the training data, which makes it heavy and rigid. By rigidity we mean here that the model often lacks a restricted number of knobs allowing mutations of the system. In that respect, global models are preferable because of their smaller size, their usually smaller computation requirements, and their limited fixed number of parameters (knobs). In the context of this paper, we will concentrate on a global approach and we will present a recursive method for the estimation of a polynomial model's parameters from the observation of a time series.

3. Global polynomial models

Let's assume we already have an idea concerning the embedding dimension d of our data. We would like to have a global parametric representation of the function f() introduced earlier in terms of the coefficients of some polynomial function. There are two obvious ways to view this problem. The first way is to try to find a polynomial expansion for f(). This assumes that we build a basis of orthonormal polynomial functions on which we will project f(). As our training data for f() will obviously not span the entire d-dimensional space, the term "orthonormal" for our basis has to be taken carefully. Indeed, our basis will have to be orthonormal with respect to some measure in the d-dimensional space that will be related to the support of our training data. This approach has been taken by Reggie Brown [7]. Given a particular

dimension d for the lag space, Brown builds recursively an orthonormal basis of polynomial functions through a Gram-Schmidt orthonormalization process. This approach requires an estimate of the observation's probability mass function which will be used to define a scalar product for the d-dimensional lag space.

Another way to view this problem is to fit a parametric form to the data and this is the approach we took in this work. In order to limit the size of our problem we will make an arbitrary decision concerning the maximum order q of that polynomial function. Our goal is now to fit a polynomial function P() of d variables and order q to our data with respect to some criteria.

$$z_n \approx P(z_{n-1}, z_{n-2}, .., z_{n-d})$$

In other words, we want to estimate the set of coefficients x_k such that:

$$z_n = \sum_{k=1}^{M} x_k f_{n,k}$$

where the $f_{n,k}$ are all the possible cross-products of our d variables and of order q or less:

$$f_{n,k} = (z_{n-1})^{b_{k,1}} (z_{n-2})^{b_{k,2}} \ldots (z_{n-d})^{b_{k,d}} \quad (\neq x_{n,j} \text{ if } j \neq k)$$

where the $b_{k,l}$ are integers such

$$\text{that } \forall k \in \{1,..,M\}, \sum_{l=1}^{d} b_{k,l} \leq q$$

Let N be the number of observations we have in our training set and let's define the following objects:

$$A = \begin{bmatrix} f_{d+1,1} & - & f_{d+1,M} \\ | & ... & | \\ f_{N,1} & - & f_{N,M} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ | \\ x_M \end{bmatrix} \text{ and } z = \begin{bmatrix} z_{d+1} \\ | \\ z_N \end{bmatrix}$$

Our problem reduces to solving the linear equation Ax=z for x. Of course, chances are that the number of training data N will be much bigger than M and this problem is ill-conditioned. At this point, one can think of pseudo-inversion and methods such as singular value decomposition. Indeed, a singular value decomposition would give us

$$A_{(N-d)\times M} = U_{(N-d)\times M} . \Sigma_{M\times M} . V^T_{M\times M}$$

$$\text{where } \Sigma \text{ diagonal and } U^T . U = V . V^T = I$$

and we could then estimate x as:

$$x = V . \tilde{\Sigma}^{-1} . U^T . z \text{ where } \left(\tilde{\Sigma}^{-1}\right)_{ii} = \begin{cases} 1/(\Sigma)_{ii} & \text{if } (\Sigma)_{ii} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

But given that N might in the order of 10000 or more, we can foretell the potential heaviness of such an approach. These computations could be simplified by noticing that the rank of A can't be any bigger than M but even then the method wouldn't posses the flexibility of an adaptive algorithm (the estimation has to be done from scratch for each new set of observations). Instead, we will now introduce a recursive method, namely a Kalman filter, that will solve our problem as we acquire new data.

3.1. Cross-products

Given a number of variables and an order, it might sound useful to compute the number M(d,q) of terms ($f_{n,1},...,f_{n,M}$) corresponding to the list of possible cross-products of order q or less. Having an idea concerning the number of these terms will tell us how the size of our problem grows with the dimension d and the order q.

We wish to count all the possible $z_1^{b_1} . z_2^{b_2} \ldots z_d^{b_d}$ such that $\forall k,\ b_k \in N$ and $\sum_{k=1}^{k=d} b_k \leq q$. This is equivalent to counting all the possible cross-products $1^{b_0 - 1} z_1^{b_1 - 1} . z_2^{b_2 - 1} \ldots z_d^{b_d - 1}$ such that:

$$\forall k,\ b_k \in N^* \text{ and}$$

$$\sum_{k=0}^{k=d} (b_k - 1) = q \quad \left(\text{i.e.} \sum_{k=0}^{k=d} b_k = q + d + 1 \right).$$

At this point, we can recall that there are $\binom{n-1}{k-1}$ ways to choose k non-zero positive integers that sum to n. Therefore a simple expression for M(d,q) is the following:

$$M(d,q) = \binom{q+d}{d}$$

Pascal's famous triangle, based on the property

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1},$$

allows us to build a table for M(d,q) very quickly.

q \ d	1	2	3	4	5
1	2	3	4	5	6
2	3	6	10	15	21
3	4	10	20	35	56
4	5	15	35	70	126
5	6	21	56	126	252
6	7	28	84	210	462
7	8	36	120	330	792

Fig 3.1 - M(d,q) table is Pascal's triangle.

4. Recursive estimation

4.1. The problem

Let's consider the following linear system:

$x(t+1) = F.x(t) + v(t)$ (1) (Evolution of parameters)
and
$z(t) = H(t).x(t) + w(t)$ (2) (Observation)

where F is a p*p matrix, H(t) is a n*p matrix and v(t) and w(t) are white noises, usually assumed to be Gaussian. In our case, we would have:

$$p=M,\ n=1,\ H(t) = \begin{bmatrix} f_{t,1} & - & f_{t,M} \end{bmatrix} \text{ and } F=I.$$

We will estimate the parameters x() given our observations of z() through the following model:

$$\hat{x}(t+1 \mid s) = F.\hat{x}(t \mid s) \quad (3)$$
and
$$\hat{z}(t \mid s) = H(t).\hat{x}(t \mid s) \quad (4)$$

Depending on the value of s with respect to t, solving that problem will accomplish different tasks:
- if s = t, we are **filtering**.
- if s < t , we are **forecasting**.
- if s > t, we are **smoothing**.

The solution that we're about to state relies on the assumption that v(t) and w(t) are centered white noises that are not correlated to x(t):

$$E[v(t)] = E[w(t)] = 0 ;$$
and
$$E\,[v(t).v^T(s)] = \partial_{ts}.Q \ ; \ E\,[w(t).w^T(s)] = \partial_{ts}.R$$

One should keep in mind that even if it might be tempting to set R and Q to be diagonal, the only thing one we can say for sure is that R and Q will be symmetric.

As for any estimation problem, we need to state clearly what our criteria is. Let's define the error covariance matrix for t knowing s as:

$$P(t \mid s) = E\,[\,(x(t) - \hat{x}(t \mid s)) \,.\, (x(t) - \hat{x}(t \mid s))^T\,]$$

Our criteria will be to minimize the trace of this matrix. One can notice that this minimization is equivalent to a least mean square criteria applied to the parameter set x().

4.2. The solution

The solution is known as a Kalman filter:

$$\hat{x}(t \mid t) = \hat{x}(t \mid t-1) + L(t) \,.\, (z(t) - \hat{z}(t \mid t-1)) \quad (5)$$

It was rigorously demonstrated that this system was optimal in the case where the different white noises introduced earlier were Gaussian. It is not our intention in this paper to present the demonstration of this result but we can provide a few hints that should at least provide a intuitive understanding for the relation (5).

Hints:

We're trying to solve the BLS in Gaussian case (i.e. LLSE). Furthermore our system is linear:

$$\Rightarrow \quad \hat{\underline{X}}_L(\underline{Z}) = \underline{m}_x + \Lambda_{xz}\Lambda_z^{-1}(\underline{Z} - \underline{m}_z)$$

Let's write : $\hat{\underline{X}}_L\begin{pmatrix}\begin{bmatrix} z_1 \\ | \\ z_M \end{bmatrix}\end{pmatrix} \stackrel{\Delta}{=} \hat{\underline{X}}(M)$

So rather than inverting growing MxM matrix each time we get a new observation, we can use Gram-Schmidt to find a recursive relationship for the estimate:

$$\Lambda_z = \Gamma.\Lambda_e.\Gamma^T \quad \text{i.e.} \quad \begin{cases} e_1 = z_1 \\ e_2 = z_2 - \gamma_{21}e_1 \ \left(\gamma_{21} = \dfrac{\langle z_2, e_1\rangle}{\langle e_1, e_1\rangle}\right) \\ \vdots \\ e_M = z_M - \hat{z}(M|M-1) \end{cases}$$

which will end up giving us a recursive relationship on the estimate:

$$\hat{\underline{X}}(i+1) = \hat{\underline{X}}(i) + \Lambda_{xe_{i+1}}\lambda_{e_{i+1}}^{-1}e_{i+1}$$

It will be the task of L(t) to make sure the trace of P(t|t) is minimized. Determining what this minimization implies on L(t) will give us its optimal expression.

From (1) and (3):
$(\,x(t) - \hat{x}(t|t-1)\,) = F \,.\, (x(t-1) - \hat{x}(t-1|t-1)) + v(t-1)$
and therefore: $P(t|t-1) = F \,.\, P(t-1|t-1) \,.\, F^T + Q \quad (6)$

Combining (1) and (5), we get:
$(\,x(t) - \hat{x}(t|t)\,) = (\,I - L(t).H\,).(\,x(t) - \hat{x}(t-1|t-1)) - L(t).w(t-1)$
i.e.
$$P(t|t) = (I - L(t).H) \,.\, P(t|t-1) \,.\, (I - L(t).H)^T + L(t) \,.\, R \,.\, L^T(t) \quad (7)$$

(7) is a quadratic equation (where the unknown is L(t), a n*p matrix). The minimization of Tr[P(t|t)] can sound complicated a priori. Yet, as for simple second degree equations, we can write (7) in its canonical form:

$$P(t \mid t) = (\,L.S - T\,) \,.\, (\,L.S - T\,)^T + M \quad (7\text{Bis})$$

As we know that P(t|t) is positive semi-definite, we are sure that M will also be positive semi-definite and the minimization of Tr[P(t|t)] falls into the zeroing of the term (L.S - T).
By identification between (7) and (7Bis), we get:

$$S.S^T = [\, R + H\,.\,P(t|t-1)\,.\,H^T] \qquad (8)$$

and

$$S.T^T = [\, H\,.\,P(t|t-1)\,] \qquad (9)$$

$$(9) \Rightarrow T^T = S^{-1}.[\, H\,.\,P(t|t-1)\,] \Rightarrow T = P(t|t-1)\,.\,H^T.\,S^{-T}$$
$$\Rightarrow L(t) = T\,.\,S^{-1} = P(t|t-1)\,.\,H^T.\,(S.S^T)^{-1}$$

So finally, using (8) we have:

$$\boxed{L(t) = P(t|t-1)\,.\,H^T.[\, R + H.\,P(t|t-1)\,.\,H^T\,]^{-1}} \qquad (10)$$

which provides us with the optimal expression for L(t).

4.3. Recursive computation of P(t|t-1)

The last step we need to introduce is a recursive relationship which will update our estimate of the error covariance matrix. Lets write P(t|t-1) as Σ(t). As we will see, the equations (6), (7) and (10) give us a simple recursion on Σ(t).

From (7),

$$P(t|t) = \Sigma(t) - \Sigma(t).H^T.L^T - L.H.\Sigma(t) + L.H.\Sigma(t).H^T.L^T + L.R.L^T$$

and from (10),

$$P(t|t) = \Sigma(t) - L.H.\Sigma(t) - \Sigma(t).H.A^{-T}H.\Sigma(t) + L.A.L^T$$

where $A = [R + H.\Sigma(t)H^T]$

R and Σ(t) are symmetric and therefore, A is too. $A^T = A$, $A^{-T} = A^{-1}$ and $\Sigma^T(t) = \Sigma(t)$. So by applying the relation (10) once again, we finally get:

$$P(t|t) = \Sigma(t) - L.H.\Sigma(t) - \Sigma(t).H^T.A^{-1}.H.\Sigma^T(t) + \Sigma(t).H^T.A^{-1}.A.A^{-1}.H.\Sigma^T(t)$$

and after simplifications we get:

$$P(t|t) = \Sigma(t) - L.H.\Sigma(t) \qquad (11)$$

So by injecting this expression for P(t|t) in the equation (6) we will finally get a recursion on Σ(t) (i.e. P(t|t-1)):

$$\boxed{\Sigma(t+1) = F\,.\,[\, \Sigma(t) - L(t).H(t).\Sigma(t)\,]\,.\,F^T + Q} \qquad (12)$$

4.4. The algorithm

By now, we have gathered all the pieces we need and by putting them back together, we will describe the recursive method that will solve our estimation problem. After having guessed some initial values, the algorithm induced by this method is provided by the relations (3), (5), (10) and (12).

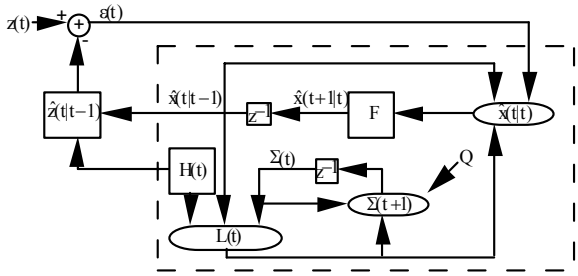
(i) $\boxed{L(t) = P(t|t-1)\,.\,H^T.[\, R + H.\,P(t|t-1)\,.\,H^T\,]^{-1}}$

(ii) $\hat{x}(t|\,t) = \hat{x}(t|\,t-1) + L(t)\,.\,(z(t) - \hat{z}(t|\,t-1))$

(iii) $\boxed{\Sigma(t+1) = F\,.\,[\, \Sigma(t) - L(t).H(t).\Sigma(t)\,]\,.\,F^T + Q}$

and $\hat{x}(t+1\,|\,t) = F.\hat{x}(t|\,t)$

and update H to H(t+1) (i.e. compute the new values of the cross products)



This system will fit a polynomial function of arbitrary dimension and order to the data without requiring the construction of an orthonormal set of functions. We recall that such an orthonormal set would have to be specific to the data as the scalar product is derived from an estimate of the data's probability mass function (PMF). Estimating such a PMF can already be a source of confusion and it is often seen as a modeling problem itself. Only then can we start building the orthonormal set of functions through Gram-Schmidt on which the data will be projected (using the same scalar product). Just as for the method we present in this paper, the choice of the estimated polynomial function's order is arbitrary, it is fixed by the number of orthonormal polynomials on which the data was projected.

5. Implementation and evaluation

The previous algorithm was implemented in C on a SGI Indigo. As our main concern was the analysis of musical instruments' timbre, our data was a set of sampled sounds.

5.1. A few words on R and Q

An insightful guess concerning the values to set Q and R to would require some advanced knowledge about the system. In some restricted linear cases, an analytical approach can provide some clues to the optimal

tuning of R and Q. Realistically, the best we can do in our case is to make reasonable assumptions. In the context of this work, the matrix Q was chosen to be diagonal. If we recall the definition of this matrix, this assumes that the parameters of our model evolve in uncorrelated fashion. The actual values of this diagonal matrix as well as the value of the scalar R can be comprehended as convergence regulators.

5.2. The data

The data we chose is a sampled audio recording of a bowed violin string. We restricted our attention to the quasi-periodic part of this recording which we normalized between -1.0 and 1.0. The figure 5.1 shows a plot of this data in a three-dimensional lag space as well as an estimate of its spectrum.

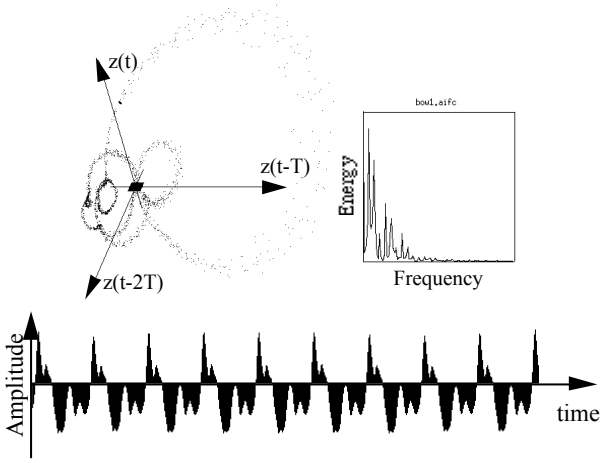


Fig 5.1 - The data

A parallel investigation of another method for the estimation of non-linear models from embeddings (computationally much more expensive) allowed the author to build a very accurate non-linear model of dimension 4 for the same data. This work suggests that the data's embedding dimension should be around 4. We can still evaluate the data's joint entropy and redundancy for various dimensions in order to get a confirmation concerning the embedding dimension of our observation. A binary tree was used in order to estimate the data's joint entropy for d = 1 to 7. This method is an efficient way to compute these entropies and its representation of the data's probability mass function is equivalent to a histogram were the data's resolution determines the width of the bins.

Figure 5.2 is a plot of the estimated entropies and redundancies of our data with a resolution of 8 bits and it reveals an embedding dimension of 4 or 5.

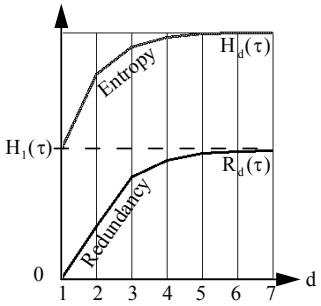


Fig 5.2 - Entropies and redundancies for increasing dimension d.

5.3. The model

As we pointed it out earlier, the algorithm requires from the user to suggest a maximum order for the polynomial function to be fitted. The Kalman filter may require a few passes through the data set in order to converge and the observation of the out-of-sample error after each pass is an indicator of its state of convergence.

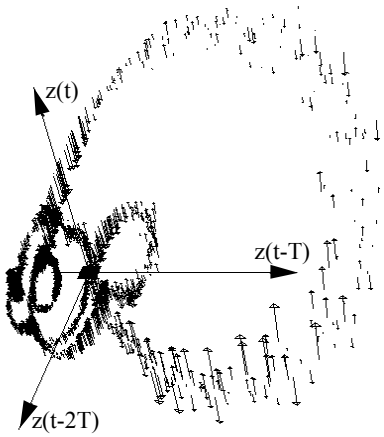


Fig 5.3 - Out-of-sample error in a 3D lag space

Figure 5.3 plots this error after a single pass through the data (the dimension of the model was set to 4 and the order of the polynomial to 4). In this 3D lag space, the arrows go from the real data to their corresponding estimate through the model.

5.4. Evaluation

The error mean is probably the most obvious quantity to observe when evaluating the performance of an estimator. The following figure shows this quantity as a function of the number of passes through our data set for different architectures for our model.

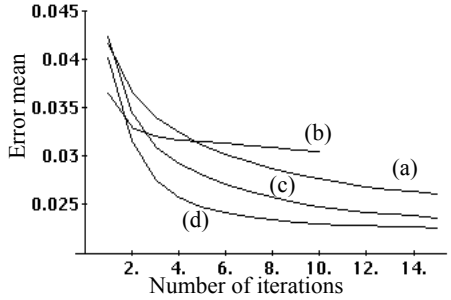


Fig 5.4 - Evolution of the error mean as a function of the number of passes through the data
(a) Model of dimension 4 and order 4.
(b) Model of dimension 5 and order 3.
(c) Model of dimension 4 and order 5.
(d) Model of dimension 5 and order 4.

A high error mean will definitely indicate a model mismatch but even if this mean is small, we are not guaranteed that the model doesn't miss some important structure in our data. As a sanity check, we might want to make sure the out-of-sample error distribution doesn't reveal any particular structure. The following plots (figures 5.5 and 5.6) are histograms of the absolute value of the out-of-sample error for different model architectures (i.e. different orders) and at different stages of convergence.

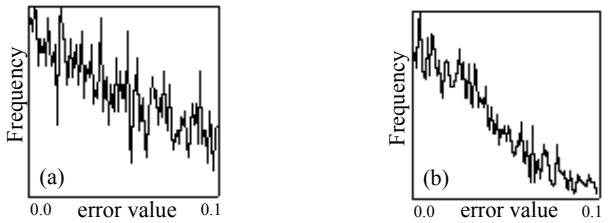


Fig 5.5 - Histogram of the prediction error for a polynomial model of dimension 4 and order 4.
(a) after a single pass through the data
(b) after 15 passes through the date

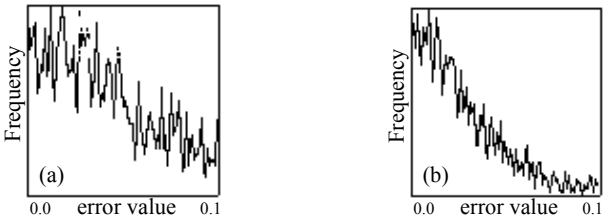


Fig 5.6 - Histogram of the prediction error for a polynomial model of dimension 4 and order 5.
(a) after a single pass through the data
(b) after 15 passes through the date

6. Future directions

An investigation of the use of radial basis function and its relationship with probability mass function estimation gave us some insights concerning means by which one could detect relevant regions of an attractor. We believe that the performance of our polynomial models will increase if they are estimated over a smaller but more relevant data set.

In the version of the estimator that we presented in this paper the criteria we chose was a simple LMS (least mean square). This straight forward criteria is traditionally a good choice for general estimation tasks but it doesn't take the specificity of our case in account. Indeed, the hypersurfaces we estimate here are the supports of attractors and in its present form, our algorithm doesn't make any use of information such as Lyapunov exponents. Building a more insightful criteria will be another one of our concerns for future work.

The use of polynomial functions was guided by the ease with which the estimation could be turned into a linear problem but we do not intend to constrain our work to this single architecture.

Finally, we are in the process of extending our investigation to non-autonomous systems. Works from Gasdagli (see [8]) suggest a generalization of theoretical results on the modeling of autonomous dynamical systems to input-output systems. Interesting systems usually have inputs and a generalization of our tools to these cases is under current study.

7. References

[1] Floris Takens. *Detecting strange attractors in turbulence*. Lecture Notes in Mathematics vol.898 (Springer, Berlin) p.366-381, 1981.

[2] Neil A. Gershenfeld. *An Experimentalist's Introduction to the Observation of Dynamical Systems*. Directions in Chaos, Vol.II, Hao Bai-lin ed., World Scientific, 1988.

[3] J.-P. Eckmann and D. Ruelle. *Ergodic theory of chaos and strange attractors*. Reviews of Modern Physics, Vol.57, No.3, Part I, July 1985.

[4] Neil A. Gershenfeld and Andreas S. Weigend. *The Future of Time Series*. Time Series Prediction: Forecasting the Future and Understanding the Past, p.1-70, Addison-Wesley, 1993.

[5] Henry D. Abarbanel, Reggie Brown, John J. Sidorowich, and Lev Sh. Tsimring. *The analysis of observed chaotic data in physical systems.* Reviews of Modern Physics, Vol.65, No.4, October 1993.

[6] Robert G. Brown and Patrick Y.C. Hwang. *Introduction to Random Signals and applied Kalman Filtering* (second edition). John Wiley & Sons, 1992.

[7] Reggie Brown. *Orthonormal Polynomials as Prediction Functions in Arbitrary Phase Space*

Dimensions. Institute for Nonlinear Science, University of California, San Diego, 1994.

[8] Martin Casdagli. *A Dynamical Systems Approach to Modeling Input-Output Systems*. Nonlinear Modeling and Forcasting, SFI Studies in the Sciences of Complexity, Proc. Vol.XII, Eds M. Casdagli & S. Eubank, Addison-Wesley, 1992.