Activity Recognition and Object Tracking Based on Multiple Models

Author: Tiago Simão
Supervisor: Miguel Barão

This work has been performed in the framework of project: ARGUS - Activity recognition and object tracking based on multiple models, supported by FCT under contract: PTDC/EEA-CRO/098550/2008.

April 27, 2012
Contents

1 Introduction 3

2 Standard Offline Solution 4
   2.1 Space discretization 4
   2.2 Vector fields prior 6
   2.3 Single Vector Field Estimation 7
   2.4 Results of Offline Standard Implementation 10

3 Recursive Vector Field Estimation 15
   3.1 Online Implementation Methods 15
      3.1.1 Recursive Least Squares 15
      3.1.2 Recursive Vector Field through Kalman Filter 17
      3.1.3 Variance of the nodes in the vector field 19
   3.2 Examples for recursive implementation 19
      3.2.1 Recursive vector field estimation 20
      3.2.2 Consecutive vector field estimation 20
      3.2.3 Trajectory prediction using vector field estimation 21

4 Vector Field Estimation for Multiple Trajectories 25
   4.1 Recursive Vector Field estimation for Multiple trajectories 27
   4.2 Examples for multiple trajectories estimation 27

5 Multiple Vector Fields with Transition Matrix 29
   5.1 Multiple Vector Field Offline Estimation 30
      5.1.1 E-Step of EM 30
      5.1.2 M-Step of EM 34
      5.1.3 Expectation-Maximization algorithm 35
   5.2 Multiple Vector Field Online Estimation 35

6 Transition Probabilities estimation 36
   6.1 Transition Probabilities Online Estimation 40
   6.2 Examples for Online estimation of Transition Matrix 41

7 Conclusions and Future Work 41

References 43

A Matrix 45

B Trace 46

C Schur / Hadamard Product 46

D Determinant 47
1 Introduction

The motion of objects has received a special attention in several research areas. In the center of that research is human motion analysis in the surveillance area [7, 1, 18]. With the purpose of identifying and studying actions of living beings or objects with motion, a trajectory can be obtained by resorting to image processing techniques [4, 1, 10, 9]. In this report, we assume that video sequences are acquired and processed to generate sample of trajectories. This trajectories are then used to identify a vector field in the image space that best describes it [11, 10, 9].

We assume that the original system generating the observed trajectory is unknown. Therefore, the vector field is obtained through the interpolation of a set of nodes in a uniform grid, where both norm and direction are needed to reproduce the trajectory. We can assume that the vector field models a nonlinear system, where we only have access to the output measurements defined as the trajectory positions. Previous works related with the identification of nonlinear systems already encompass a variety of approaches [8, 14, 15, 19, 12]. The algorithms used to estimate parameters include expectation-maximization [17] and particle filters [5, 16].

Trajectory analysis based on vector fields has already been performed by [10, 9]. However, parameter estimation is done off-line and the update of a new sample requires the computation of the complete trajectory over again. Here, we first present the off-line approach as the standard solution and then describe the recursive parameter estimation solution, which can be obtained from the former. When a new position is added to the trajectory, the previous trajectory information must be stored to compute the estimation of the vector field, but without computing all the trajectory positions already processed.

The main contribution of this work is the recursive version of the Bayesian algorithm developed in [10]. Our algorithm achieves the same results by updating a sufficient statistic of the past data. This update is done in constant time and memory complexity and avoids some matrix inversions that appear in the original problem.

To turn the standard method to an online algorithm, we go through a step-by-step problem formulation, from a simpler problem until we end up with the initial problem. In section 2 we present the approach method to the object motion problem including the space discretization and prior selection. Section 3 presents the recursive implementation using the recursive least squares algorithm for a single trajectory and a single vector field. Next in section 4 we extend the recursive implementation to multiple trajectories. Sections 5 and 6 explore the estimation of multiple vector fields and a single transition matrix using the expectation maximization algorithm and how to implement a recursive version of the algorithm. In section 7 we present
some ideas for future work.

2 Standard Offline Solution

Our objective is to estimate a vector field that best describes an observed trajectory. We consider that each trajectory position is given by

\[ x_t = x_{t-1} + T(x_{t-1}) + w_t, \]  

(1)

where \( x_t \in \mathbb{R}^{D \times 1} \) is the position with dimension \( D \) at instant \( t \) and \( T(x_t) \in \mathbb{R}^{D \times 1} \) is the velocity vector (or step) at position \( x_t \). We assume \( w(t) \sim \mathcal{N}_D(0, \Sigma_t) \) as a zero-mean multivariable gaussian with covariance matrix \( \Sigma_t \).

We can represent the current position in the trajectory, given the previous position, as the conditional probability

\[
p(x_t | x_{t-1}) = \mathcal{N}_D(x_t | x_{t-1} + T(x_{t-1}), \Sigma_t)
\]

(2)

where \( |\Sigma_t| \) is the determinant of the matrix \( \Sigma_t \) and where \( \|X\|_A^2 \triangleq \text{Tr}(X^TAX) \) with \( X \in \mathbb{R}^{m \times n} \) and \( A \in \mathbb{R}^{m \times m} \) for any positive integer \( m,n \). The form \( \text{Tr}(B) \) means the trace of the square matrix \( B \). Equation (2) can also be referred to as the step probability distribution.

Consider that \( X = [x_0, x_1, \ldots, x_L] \in \mathbb{R}^{D \times (L+1)} \) contains all \( L + 1 \) positions of a single trajectory. The joint probability of the complete trajectory is

\[
p(X_L) = p(x_0, x_1, \ldots, x_L) = p(x_0) p(x_1 | x_0) \ldots p(x_L | x_{L-1})
\]

(3)

\[
= p(x_0) \prod_{t=1}^{L} p(x_t | x_{t-1}),
\]

where \( p(x_0) \) is the probability distribution of the initial position.

The problem here presented is similar to the Gaussian process [20] and some other interpretations and implementations could be used for its solution.

2.1 Space discretization

As we cannot compute the velocity vector \( T(x_t) \) for an infinite number of positions in the D-dimensional space, we need to perform a space discretization. Therefore, the space is discretized into a finite grid and a set of velocity vectors \( t_n \) are associated to the nodes of the grid. We then assume
that the velocity vector $T(x)$ is obtained by interpolation

$$T(x_{t-1}) \triangleq \sum_{n=1}^{N} t_n \phi_n(x_{t-1}) , \quad (4)$$

where $N$ is total number of nodes, $t_n \in \mathbb{R}^{D \times 1}$ is the velocity vector at node $n$ and $\phi_n(x_t) \in \mathbb{R}$ is a normalized weight function

$$\phi_n(x_t) \triangleq \frac{w_n(x_t)}{\sum_{j=1}^{N} w_j(x_t)} , \quad (5)$$

which satisfies the constraints

$$0 \leq \phi_n(x_t) \leq 1 , \quad \sum_{n=1}^{N} \phi_n(x_t) = 1 , \quad (6)$$

and where the weights

$$w_n(x_t) \triangleq (d(x_t, g_n) + d_0)^{-p} , \quad (7)$$

represent the inverse of the distance $d(x_t, g_n)$ from position $x_t$ to node $g_n$. The parameter $p > 0$ adjusts the smoothness and $d_0 > 0$ is a small additive constant to avoid an infinite weight $w_n$ when $x_t = g_n$.

Equation (4) can be rewritten as

$$T(x_{t-1}) = T \Phi_x t , \quad (8)$$

where the matrices $T \in \mathbb{R}^{D \times N}$ and $\Phi_x t \in \mathbb{R}^{N \times 1}$ are defined by

$$T \triangleq [t_1, t_2, \ldots, t_N] , \quad \Phi_x t \triangleq \begin{bmatrix} \phi_1(x_{t-1}) \\ \phi_2(x_{t-1}) \\ \vdots \\ \phi_N(x_{t-1}) \end{bmatrix} . \quad (9)$$

We define the velocity at position $x_t$ as $v_{x_t} \triangleq x_t - x_{t-1}$ and use equation (8) to represent the multivariable Gaussian distribution of equation (2) as

$$p(x_t | x_{t-1}) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_t|}} e^{-\frac{1}{2} \|v_{x_t} - T \Phi_x t\|^2_{\Sigma_t^{-1}}} . \quad (10)$$

where $T$ parameterizes the vector field we want to estimate.

The estimation of matrix $T$ is done within a Bayesian framework. In this framework an observation model and a prior distribution are required to produce a posterior distribution of $T$ given the observed trajectory. We use equation (10) as the observation model, and then define a prior distribution $p(T)$ for the vector field.
2.2 Vector fields prior

Since there is no prior information on particular directions of the vector field, the prior distribution does not convey that type of information. However, we can state that the vector field has some degree of smoothness so that neighboring nodes follow similar directions.

We represent the vector field prior by $p(T)$. Let $\mathcal{I}$ denote the set of pairs of indices $(i, j)$ containing neighboring nodes, i.e.,

$$\mathcal{I} = \{(i, j) | i \text{ and } j \text{ are neighbors, and } i \neq j\}.$$  

In a 2-dimensional space, we choose the neighbors of a node as the closest ones in all vertical, horizontal and diagonal directions. Therefore, a node in the middle of a 2-dimensional grid has 8 neighboring nodes, in the border it has 5 neighboring nodes and in a corner it has 3 neighboring nodes.

Assuming that neighboring nodes have similar directions, the prior is defined as a multivariable Gaussian

$$p(T) \propto e^{-\frac{1}{2\rho} \sum_{(i,j) \in \mathcal{I}} \| t_i - t_j \|^2}. \quad (11)$$

Letting $\Delta \in \{-1, 0, 1\}^{N \times (\#\mathcal{I})}$ denote the matrix that operates the differences between neighbors\(^1\), we obtain

$$T \Delta = [t_1, t_2, \ldots, t_N] = [t_1 - t_2, t_1 - t_3, \ldots, t_2 - t_3, t_2 - t_4, \ldots, t_{N-1} - t_N],$$

where each element $t_i - t_j$ measures the change of velocity between two neighboring nodes $(i, j) \in \mathcal{I}$.

The exponent in equation (11) can be written as a Frobenius norm on $T^\top$ as\(^2\)

$$\sum_{(i,j) \in \mathcal{I}} \| t_i - t_j \|^2 = \text{Tr} \left( (T \Delta)^\top (T \Delta) \right) = \text{Tr} \left( (T \Delta)(T \Delta)^\top \right) = \text{Tr} \left( T \Delta \Delta^\top T^\top \right) = \| T \Delta \|^2. \quad (13)$$

\(^1\)See Appendix K how we obtain the neighbor matrix.
\(^2\)Equation (13) is achieved based on trace properties and the definition of $\| X \|^2_\Lambda$ in section 2 and appendix B.
where $\Lambda$ is a symmetric and singular matrix.

Therefore, the prior multivariable Gaussian can be represented by

$$p(T) \propto e^{-\frac{1}{2\rho} \|T\|_\Lambda^2}.$$  \hfill (14)

The parameter $\rho$ is adjusted manually, according to the complexity of a trajectory, and it tunes the dependency of neighboring nodes, thereby controlling the smoothness of the vector field.

### 2.3 Single Vector Field Estimation

Our objective is to estimate the vector field parameterized by matrix $T = \{T\}$, given a set of positions of a trajectory $X$. Applying the Bayes’ law, we get to the posterior distribution

$$p(T | X) = \frac{p(X | T) p(T)}{C},$$ \hfill (15)

where $C = \frac{1}{p(X)}$ is a normalization factor, $p(X | T)$ is the trajectory joint probability model obtained in (3) and $p(T)$ is the prior defined in (14).

For estimation purposes, we use the maximum a posteriori (MAP) criterion, i.e., we must find the parameters $T$ that maximize $p(T | X)$

$$\hat{T} = \arg \max_T p(T | X) = \arg \max_T \log p(T | X)$$

$$= \arg \max_T \left( \log C + \log p(X | T) + \log p(T) \right)$$

$$= \arg \max_T \left( \log p(X | T) + \log p(T) \right).$$ \hfill (16)

To solve equation (16) we find the stationarity points with

$$\frac{\partial \log p(T | X)}{\partial T} \bigg|_{T=\hat{T}} = 0.$$

Since $\log p(T | X)$ is concave, the unique stationarity point found is the global maximum. Separately considering each term in (16), we can discard the first, $\log C$, since it does not depend on $T$.

The second parcel can be obtained with the logarithm of equation (3) and using equation (10), resulting in

$$\log p(X | T) = \log p(x_0) + \sum_{t=1}^{L} \log p(x_t | x_{t-1})$$

$$= \log p(x_0) - \frac{1}{2} \sum_{t=1}^{L} \log(2\pi)^D |\Sigma_t|$$

$$- \frac{1}{2} \sum_{t=1}^{L} \|v_{xt} - T\Phi_{xt}\|_{\Sigma_t^{-1}}^2 .$$ \hfill (18)
We assume a constant disturbance covariance matrix $\Sigma = \Sigma_t$ and define $\Phi_X \in \mathbb{R}^{N \times L}$ and $v_X \in \mathbb{R}^{D \times L}$ by

$$
\Phi_X \triangleq [\Phi_{x_1}, \Phi_{x_2}, \ldots, \Phi_{x_L}]
$$

$$
v_X \triangleq [v_1, v_2, \ldots, v_{x_L}].
$$

Performing some calculations based on matrix trace properties (shown in Appendix I), we get to

$$
\log p(X|T) = \log p(x_0) - \frac{L}{2} \log(2\pi)^{D} |\Sigma| - \frac{1}{2} \|v_X - T\Phi_X\|_{\Sigma^{-1}}^2.
$$

which resembles a least squares form.

The prior parcel from equation (16) can be obtained with the logarithm of equation (14)

$$
\log p(T) = \xi - \frac{1}{2p} \|T\|_{\Lambda}^2
$$

$$
= \xi - \frac{1}{2p} \text{Tr}(T\Lambda T^T).
$$

where $\xi$ is a constant independent from $T$.

Computing the derivatives separately [3] yields

$$
\frac{\partial}{\partial T} \log p(X|T) = \frac{\partial}{\partial T} \left( -\frac{1}{2} \|v_X - T\Phi_X\|_{\Sigma^{-1}}^2 \right)
$$

$$
= \Phi_X (v_X - T\Phi_X)^T \Sigma^{-1}
$$

$$
= \Phi_X v_X^T \Sigma^{-1} - \Phi_X \Phi_X^T T^T \Sigma^{-1},
$$

and

$$
\frac{\partial}{\partial T} \log p(T) = -\frac{1}{p} \Lambda T^T.
$$

Therefore, the result is

$$
\frac{\partial}{\partial T} \log p(T|X) = \frac{\partial}{\partial T} \log p(X|T) + \frac{\partial}{\partial T} \log p(T)
$$

$$
= \Phi_X v_X^T \Sigma^{-1} - \Phi_X \Phi_X^T T^T \Sigma^{-1} - \frac{1}{p} \Lambda T^T,
$$

which replacing it into equation (17) and transposing yields

$$
-\Sigma^{-T} T \Phi_X \Phi_X^T - \frac{1}{p} T \Lambda + \Sigma^{-T} v_X \Phi_X^T = 0
$$

$$
\Leftrightarrow -\Sigma^{-T} A_{T} \Phi_X \Phi_X^T \Lambda^{-1} - T + \Sigma^{-T} v_X \Phi_X^T \Lambda^{-1} = 0,
$$

\[3\text{We use the derivative in equation (164) in appendix G.}\]
which is a linear equation in \( T \) of the Sylvester type and can be solved by the *Matlab* function 'dlyap'. In this case, since \( \Lambda \) is singular, we make

\[
\Lambda = \epsilon I_N + \Delta \Delta^T, \tag{26}
\]

for some \( \epsilon > 0 \) and with \( I_N \in \mathbb{R}^{N \times N} \).

If we further assume that \( \Sigma = \sigma I_D \), where \( I_D \in \mathbb{R}^{D \times D} \) is the identity matrix, we achieve

\[
\begin{align*}
- \frac{1}{\sigma} T \Phi_X \Phi_T^T - \frac{1}{\rho} T \Lambda + \frac{1}{\sigma} v_X \Phi_T^T &= 0 \\
\Leftrightarrow -T \left( \frac{1}{\sigma} \Phi_X \Phi_T^T + \frac{1}{\rho} \Lambda \right) + \frac{1}{\sigma} v_X \Phi_T^T &= 0 \\
\Leftrightarrow T &= v_X \Phi_T^T \left( \Phi_X \Phi_T^T + \frac{\sigma}{\rho} \Lambda \right)^{-1}.
\end{align*}
\tag{27}
\]

Equation (27) defines the standard batch vector field estimate for a trajectory of length \( L \). As the trajectory length increases, so do the dimensions of matrices \( v_X \) and \( \Phi_X \) increase, along with the time complexity to estimate \( T \). Therefore, this solution can only be used in constrained problems.
2.4 Results of Offline Standard Implementation

We present in figure 1 four trajectories examples, where (a), (b) and (c) are 2-dimensional and (d) is 3-dimensional.

Figure 1: Example trajectories in 2-dimensions (a), (b) and (c) and in 3–dimensions (d). Trajectories (a), (b) and (c) evolve from left to right, in $x$ axis, and (d) evolves from bottom to top, in $z$ axis.
We can test the parameters $p$, $\sigma/\rho$, and the number of nodes $N$ using equation (27). We also perform a simulation of a trajectory starting at $x_0$ as the trajectories in figure 1. The standard parameters are $p = 4$, $\sigma = 1$, $\rho = 10$ and $N = 15 \times 15$ for 2-dimensions and $N = 3 \times 3 \times 10$ for 3-dimensions. The best results obtained are shown in figure 2 for each example, where we can see the proximity between the initial trajectory (in blue) and the simulated trajectory (in red).

Figure 2: Trajectories of figure 1 with vector fields in green and a model trajectory in red.
Let's start with changing the parameter $p$ that adjusts the weight of the nodes to a trajectory point by the distance between them. We can observe in figure 3 that a higher value of $p$ creates a vector field more homogenous. The interval $p = [4, 10]$ leads to the better fits of the simulation for this trajectory.

Figure 3: Trajectories of figure 1 with vector fields in green and a simulated trajectory in red when $p$ is changed.
If we change the number of nodes $N$, we get figure 4 where we can see that the more number of nodes drives to a better simulated trajectory performance and a better vector field appearance. However, we need a number of nodes computationally reliable for further solutions implementations.

Figure 4: Trajectories of figure 1 with vector fields in green and a model trajectory in red when the grid is changed.
In figure 5 we adjust the parameter $\rho$ where we can see that the greater its values, the better is the model fitting, but if the value is too large, the vector fields became less smooth. The parameter $\rho$ adjusts the neighboring variance between nodes and for low $\rho$ we get similar neighboring vectors.

![Multiple Curves with $\rho = 0.1$](image1)

![Multiple Curves with $\rho = 1$](image2)

![Multiple Curves with $\rho = 10$](image3)

![Multiple Curves with $\rho = 10$](image4)

Figure 5: Trajectories of figure 1 with vector fields in green and a simulated trajectory in red when $\rho$ is changed.
3 Recursive Vector Field Estimation

In this section, we propose a recursive version of the algorithm that does not suffer from the time and memory complexity of the batch mode version. Specifically, we seek an algorithm that has constant memory and computational complexity. These objectives can be achieved by using a sufficient statistic of the data that is updated at each step as new data is acquired.

3.1 Online Implementation Methods

We present two methods to find a recursive implementation for the problem presented. The first implements the recursive least squares method [6], which uses past information as prior for the vector field update given a new position of the trajectory. The second is based on the Kalman filter. Both ways end up with the same solution.

3.1.1 Recursive Least Squares

In the first method, we start by rewriting equation (3) as

\[ p(X_t) = p(x_0, x_1, \ldots, x_t) = p(x_0) p(x_1 | x_0) \ldots p(x_t | x_{t-1}) = p(x_0) \prod_{i=1}^{t-1} p(x_i | x_{i-1}) p(x_t | x_{t-1}) , \]

(28)

where \( p(x_{t-1}) \) is a known joint probability of the complete trajectory on the previous time instant \( t - 1 \). The probability for the new step \( p(x_t | x_{t-1}, T_t) \) follows equation (10). Therefore, the Bayes’ law can be represented as

\[ p(T_t | X_t) = C_t p(X_t | T_t) p(T_t) \]

(29)

where now the prior includes all the previous information.

The stationary condition (17) can now be rewritten, using equations (22) and (23), as

\[
\frac{\partial}{\partial T_t} \log p(T_t | X_t) = \frac{\partial}{\partial T_t} \log p(x_t | x_{t-1}, T_t) + \frac{\partial}{\partial T_t} \log p(X_{t-1} | T_t) p(T_t) \\
= \Phi_{x_t} v_{x_t} \Sigma_{x_t}^{-1} - \Phi_{x_t} \Phi_{x_t}^T T_t \Sigma_{x_t}^{-1} \\
+ \Phi_{X} v_{X}^T \Sigma^{-1} - \Phi_{X} \Phi_{X}^T T_t \Sigma^{-1} - \frac{1}{\rho} \Lambda T_t,
\]

(30)

where now \( \Phi_{X} \) and \( v_{X} \) correspond to data obtained before time \( t \).
Replacing again into equation (17) and assuming that $\Sigma = \Sigma_t = \sigma I_D$, we get

$$
\frac{1}{\sigma} v_{x_t} \Phi^T_{x_t} + \frac{1}{\sigma} v_X \Phi^T_X = T_t \left( \frac{1}{\sigma} \Phi_{x_t} \Phi^T_{x_t} + \frac{1}{\sigma} \Phi_X \Phi^T_X + \frac{\rho}{\sigma} \Lambda \right)
$$

$$
\Leftrightarrow T_t = (v_{x_t} \Phi^T_{x_t} + v_X \Phi^T_X) \left( \Phi_{x_t} \Phi^T_{x_t} + \Phi_X \Phi^T_X + \frac{\rho}{\sigma} \Lambda \right)^{-1},
$$

which can become a recursive algorithm by having

$$
B(t) = v_{x_t} \Phi^T_{x_t} + B(t-1), \quad B(t-1) = v_X \Phi^T_X, \quad B(0) = 0,
$$

$$
A(t) = \Phi_{x_t} \Phi^T_{x_t} + A(t-1), \quad A(t-1) = \Phi_X \Phi^T_X, \quad A(0) = 0,
$$

where $A(t) \in \mathbb{R}^{N \times N}$ and $B(t) \in \mathbb{R}^{D \times N}$ are two matrices that store past information needed in the next iteration. The estimated vector field at instant $t$ is given by

$$
T_t = B(t) \left( A(t) + \frac{\sigma}{\rho} \Lambda \right)^{-1}. \quad (33)
$$

Moreover, since the denominator $A(t) + \frac{\sigma}{\rho} \Lambda$ appears in every iteration, we can simplify the algorithm to

$$
T_t = B(t) A'(t)^{-1}, \quad (34)
$$

using the initialization $A'(0) = \frac{\sigma}{\rho} \Lambda$. However, both values of $A'(t), B(t)$ can overflow, $A'(t \to \infty) = \infty$ and $B(t \to \infty) = \infty$, as the trajectory grows.

The algorithm can be further improved by avoiding the explicit inversion of $A'(t)$. For this, we rewrite equation (34) as a recursive least squares algorithm

$$
K_t = \left[ 1 + \Phi^T_{x_t} P_{t-1} \Phi_{x_t} \right]^{-1} \Phi^T_{x_t} P_{t-1},
$$

$$
T_t = T_{t-1} + (v_{x_t} - T_{t-1} \Phi_{x_t}) K_t,
$$

$$
P_t = P_{t-1} - P_{t-1} \Phi_{x_t} K_t,
$$

where $K_T(t)$ is known as the Kalman gain and $P$ is the covariance matrix of the estimation error [13, 2]. Equation (35) starts at $t = 2$ and $T(1)$ is initialized with equation (34) and $P(1) = A(1)^{-1}$, since it is not possible to calculate $P(0) = A(0)^{-1} \propto \Lambda^{-1}$ because $\Lambda$ is singular. In the algorithm (35), the information needed in the next iteration is stored in $T(t-1)$ and $P(t-1)$, meeting the sufficient statistics condition. Another approach to start the algorithm is to use $P(0) = (\epsilon I_N + \Lambda)^{-1}$, with $\epsilon \approx 10^{-10}$, which can improve speed with the cost of some precision degradation.

It is possible to verify that the resulting estimated vector field from equation (35) is the same as the one obtained in equation (27).

---

4See appendix L.
3.1.2 Recursive Vector Field through Kalman Filter

For the second method we assume that the nonlinear system follows the state space model

\[ T_t = T_{t-1}^T, \]
\[ v_t = T_t \Phi_t + w_t \]

(36)

where \( T_t \in \mathbb{R}^{D \times N} \) is the state to estimate, \( v_t = x_t - x_{t-1} \in \mathbb{R}^{D \times S} \) is the observed output, \( \Phi_t = \Phi_{x_t} \in \mathbb{R}^{N \times S} \) is the weight function in equation (9).

The error \( w_t \sim \mathcal{N}_D(0, \Sigma) \) follows a zero-mean multivariable gaussian where the covariance matrix is the expected value \( \Sigma = E\{w_tw_t^T\} = \sigma I_D \).

For prediction, we assume

\[ \hat{T}_t = \hat{T}_{t-1}^T, \]
\[ \hat{v}_t = \hat{T}_{t-1} \Phi_t, \]

(37)

where the prediction error and prediction covariance matrix yields

\[ e_t = T_t^T - \hat{T}_t^T = T_t^T - \hat{T}_{t-1}^T = e_{t-1}, \]
\[ P_t = E\{e_t e_t^T\} = E\{e_{t-1} e_{t-1}^T\} = P_{t-1}, \]

(38)

with \( e_{t-1} \in \mathbb{R}^{N \times D} \) and \( P_{t-1} \in \mathbb{R}^{N \times N} \) are the estimation error and covariance matrix, respectively.

For the estimation, we need to use three tricks. The first trick is to have \( \hat{T}_t \) in the form

\[ \hat{T}_t = \hat{T}_{t-1} + (v_t - \hat{v}_t)K_t \]

(39)

where \( K_t \in \mathbb{R}^{S \times N} \) is the Kalman gain.

Then we compute the estimation error

\[ e_t = T_t^T - \hat{T}_t^T \]
\[ = T_t^T - (\hat{T}_{t-1} + (v_t - \hat{v}_t)K_t)^T \]
\[ = T_t^T - \hat{T}_{t-1}^T - K_t^T(v_t - \hat{v}_t)^T \]
\[ = T_t^T - \hat{T}_{t-1}^T - K_t^T(T_t \Phi_t + w_t - \hat{T}_{t-1} \Phi_t)^T \]
\[ = T_t^T - \hat{T}_{t-1}^T - K_t^T(T_t \Phi_t + w_t - \hat{T}_{t-1} \Phi_t)^T \]
\[ = T_t^T - T_{t-1}^T - K_t^T(T_t \Phi_t + w_t - \hat{T}_{t-1} \Phi_t) \]
\[ = T_t^T - T_{t-1}^T - K_t^T(T_t \Phi_t + w_t - \hat{T}_{t-1} \Phi_t) \]
\[ = e_{t-1} - K_t^T(e_{t-1} + w_t) \]

(40)
and the covariance matrix

\[ P_t = E\{e_t e_t^T\} \]

\[ = E\left\{ [e_{t-} - K_t^T(\Phi_t^T e_{t-} + w_t^T)] [e_{t-} - K_t^T(\Phi_t^T e_{t-} + w_t^T)]^T \right\} \]

\[ = E\left\{ [e_{t-} - K_t^T(\Phi_t^T e_{t-} + w_t^T)] [\Phi_t^T e_{t-} + w_t^T] \right\} \]

\[ = E\left\{ e_{t-} e_{t-}^T - e_{t-}^T \Phi_t K_t - e_{t-} w_t K_t - K_t^T \Phi_t e_{t-} - e_{t-}^T + 
+ K_t^T \Phi_t e_{t-}^T - \frac{1}{2} \Phi_t K_t + K_t^T e_{t-} e_{t-}^T - K_t^T w_t e_{t-} + 
+ K_t^T w_t e_{t-} + w_t^T e_{t-} \cdot K_t + K_t w_t^T w_t K_t \right\}. \]

If we assume that \( E\{e_{t-} w_t\} = 0 \) and assume \( E\{w_t^T w_t\} = \sigma I_S \) (second trick), we can continue

\[ P_t = P_{t-} - P_{t-} \Phi_t K_t + K_t^T \Phi_t^T P_{t-} + K_t^T \Phi_t^T P_{t-} K_t + K_t^T \sigma K_t \]

\[ = P_{t-} - 2P_{t-} \Phi_t K_t + K_t^T [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S] K_t. \]  

(42)

The last trick is to find the optimal \( K_t^* \) gain that minimizes \( P_t \) by differentiating \( P_t \) in order to \( K_t \).

\[ \frac{\partial P_t}{\partial K_t} = \frac{\partial}{\partial K_t} \left( P_{t-} - 2P_{t-} \Phi_t K_t + K_t^T [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S] K_t \right) \]

\[ = -2P_{t-} \Phi_t + 2K_t^T [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S], \]

which transposing and equaling to zero yields

\[ -2P_{t-} \Phi_t + 2K_t^T [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S] = 0 \]

\[ \iff K_t^* = P_{t-} - 2P_{t-} \Phi_t + \sigma I_S \]

\[ \iff K_t^* = [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S]^{-1} \Phi_t^T P_{t-}. \]  

(44)

We can substitute \( K_t^* \) in \( P_t \) to get

\[ P_t = P_{t-} - 2P_{t-} \Phi_t K_t + K_t^T [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S] K_t \]

\[ = P_{t-} - 2P_{t-} \Phi_t K_t + P_{t-} [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S]^{-1} [\Phi_t^T P_{t-} - \Phi_t + \sigma I_S] K_t \]

\[ = P_{t-} - 2P_{t-} \Phi_t K_t + P_{t-} \Phi_t K_t \]

\[ = P_{t-} - P_{t-} \Phi_t K_t, \]  

(45)

because \( P_t \) and all elements in it are symmetric, we can simplify \( P_{t-} \Phi_t K_t = (P_{t-} \Phi_t K_t)^T = K_t^T \Phi_t^T P_{t-}. \)

The final algorithm is given by

\[ K_t = [\Phi_t^T P_{t-} + \sigma I_S]^{-1} \Phi_t^T P_{t-}, \]

\[ \tilde{T}_t = \tilde{T}_{t-1} + (v_t - \tilde{T}_{t-1} \Phi_t) K_t, \]

\[ P_t = P_{t-} - P_{t-} \Phi_t K_t, \]  

(46)

with is the same as equation (35).
3.1.3 Variance of the nodes in the vector field

It is possible to represent the uncertainty associated with each node of the grid through the diagonal of the covariance matrix, from equation (35), as

$$P_t = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_1 \sigma_N \\ \vdots & \ddots & \vdots \\ \sigma_N \sigma_1 & \cdots & \sigma_N^2 \end{bmatrix}, \quad \text{or} \quad \text{diag}(P_t) = [\sigma_1^2, \ldots, \sigma_N^2].$$

(47)

The uncertainty is directly related with the nodes variance which by itself is related to $\rho$ and $\Lambda$. That means that if $\rho$ increases, the relation between neighbors decreases, and the variance and uncertainty increases.

An example the uncertainty along the vector field is shown in figure 6, where we have a gray image with the colors from white to black that represent the more to less certainty in the vector field. The coners are more uncertain because they are farther away from the observed data.

![Figure 6: Uncertainty of each node for the vector field estimation.](image)

3.2 Examples for recursive implementation

To illustrate the proposed algorithm, three simulations were performed. In the first one, A, we estimate the vector field for different instants of a trajectory. In the second, B, we compare the estimated vector field using a single or two consecutive distinct trajectories. At the third one, C, we compare the performance of using a single or both trajectories to predict the future trajectory steps. The standard parameters we choose for all examples are $p = 4, \kappa = 1, N = 15 \times 15$ and $\sigma = 100$. 
We generate synthetic trajectories using the Van der Pol oscillator

\begin{align*}
    x_1(t + 1) &= x_2(t)dt + x_1(t) \\
    x_2(t + 1) &= \left(\mu(1 - x_1^2) x_2(t) - x_1(t)\right)dt + x_2(t),
\end{align*}

(48)

where we assume \( v_x = [x_1(t + 1) - x_1(t), x_2(t + 1) - x_2(t)]^T \), \( \mu = 0.01 \) and \( dt = 0.1 \). The system chosen has a limit cycle to which every trajectory converges. A trajectory outside that limit cycle has the behavior of a stable system, and a trajectory inside that limit cycle has the behavior of an unstable system. Every figure presented has a black-white squared background that represents the variance of each node, obtained with matrix \( P(t) \) from equation (35). The less variant nodes are white, and the more variant ones are black.

### 3.2.1 Recursive vector field estimation

Experiment A consists in the estimation of a vector field from the same trajectory at different time instants. In figure 7, we present the evolution of the vector field obtained from a trajectory outside the limit cycle for the instants \( t_1 = 25, t_2 = 40, t_3 = 100 \) and \( t_4 = 1001 \), which are shown in plots 7a, 7b, 7c and 7d, respectively.

Each plot shows the trajectory generated up to the specified time (blue). It also shows the estimated vector field (green) and a simulated trajectory (red) computed with the estimated vector field and the first position of the generated trajectory. Every plot in figure 7 can also be obtained in a batch mode using equation (27).

### 3.2.2 Consecutive vector field estimation

Experiment B compares the vector field estimation using a single trajectory with a pair of consecutive trajectories, each of them in different sides of the limit cycle. For that purpose, we estimate a vector field from the first trajectory and then use it as prior for the estimation of an improved vector field. We show in figure 8 the estimated vector field and the inside simulated trajectory for four possible cases, which are shown in plots 8a, 8b, 8c and 8d. The plots 8a and 8b present the vector field estimation using the trajectories inside and outside the limit cycle, respectively. Plot 8c presents the vector field estimation using the inside trajectory followed by the outside one, and 8d presents the same as 8c with the trajectories order switched.

We can verify that the simulated trajectories in 8a, 8c and 8d fit acceptably the generated trajectories. Unlike the other plots, the simulated trajectory of plot 8b (estimation using the outside trajectory) does not fit the generated trajectory. The results mentioned can be explained by the lack of information of the inside of the limit cycle in 8b.
3.2.3 Trajectory prediction using vector field estimation

In experiment C, we concern the performance of the estimated vector field to predict the future steps of a trajectory, for two estimation examples. The vector field estimation follows the same idea as in section 3.2.2, but here we only use the first 200 steps of a trajectory. Then, we present in figure 9 the generated and simulated trajectories, both with 1001 steps. Four possible cases are shown in plots 9a, 9b, 9c and 9d.

The vector field estimation using both inside and outside trajectories achieves better results than using just a single sided trajectory. These results meet the conclusions in 3.2.2, where the use of two distinct trajectories
Figure 8: Vector field estimation using trajectories with length of 1001 steps. Plots 8a and 8b use an inside and outside trajectory respectively. Plot 8c presents the vector field estimation using the inside trajectory followed by the outside one, and in 8d presents the same as 8c with the trajectories order switched.

yields acceptable model identification.

To compare the vector field model in each figure, we present in table 1 the squared mean error (MSE) of $w_t$ in equation (1). The first line of table 1 corresponds to the experiment of section 3.2.1 and each column to each plot (a) to (d) of figure 7. The second line of table 1 corresponds to the experiment of section 3.2.3 and each column to each plot (a) to (d) of figure 9. The MSE values are obtained from the difference between the steps of the generated trajectory $v_{x_t}$ and the estimated velocity vectors $T(x_t)$, computed with equation (8). Note that in table 1, we only compare
Figure 9: Vector field estimation using trajectories with length of 200 steps and simulation of a trajectory with length of 1001 steps. Plots 9a and 9b use an inside and outside trajectory respectively. Plot 9c presents the vector field estimation using the inside trajectory followed by the outside one, and in 9d presents the same as 9c with the trajectories order switched.

The results for experiment A show that the MSE does not increase with the trajectory length as expected. Instead, there is a tradeoff between the length and the space distribution of a trajectory, in order to obtain better model accuracy.

Moreover, the trajectory prediction (experiment C) using two trajectories of each side of the limit cycle presents a lower MSE than using two trajectories of the same side. In addition, we find that the results of (c) and (d)
Table 1: Mean squared error (MSE) for experiences A and C corresponding to sections 3.2.1 and 3.2.3. Each table line corresponds to the figures 1 and 3, or experiments A and C, where each column corresponds to each figure plot (a) to (d).

<table>
<thead>
<tr>
<th></th>
<th>MSE</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp. A</td>
<td>6.33e-7</td>
<td>7.38e-7</td>
<td>6.29e-7</td>
<td>7.74e-7</td>
<td></td>
</tr>
<tr>
<td>Exp. B</td>
<td>9.37e-7</td>
<td>1.29e-6</td>
<td>9.58e-7</td>
<td>9.58e-7</td>
<td></td>
</tr>
<tr>
<td>Exp. C</td>
<td>4.86e-5</td>
<td>1.94e-6</td>
<td>1.73e-6</td>
<td>1.73e-6</td>
<td></td>
</tr>
</tbody>
</table>

of experiment C are the same. From this, we can conclude that the order in which each trajectory is computed is not relevant. On the other hand, we can verify that the model identification, for the present system, shows promising results for the prediction of future trajectory steps.
4 Vector Field Estimation for Multiple Trajectories

In the case where we have multiple trajectories, each starting in different time instants, the theory will be slightly different from the one in section 2.3. We now need to estimate $T$ given the set of trajectories $X = \{X_1, \cdots, X_S\}$, each can have different length $X_s = \{x_{s0}, \cdots, x_{tL_s}\}$ and start at different instants $t_s$. The probability distribution is

\[ p(T \mid X) = C_X p(X \mid T) p_X(T), \quad (49) \]

where we consider independent trajectories to have

\[ p(X \mid T) = \prod_{s=1}^S p(X_s \mid T). \quad (50) \]

We need to choose a prior for equation (49) like we did before. As we already have the prior for one trajectory, we can choose

\[ p_X(T) = \prod_{s=1}^S p(T). \quad (51) \]

and also $C_X = \prod_{s=1}^S C$, which leads to

\[ p(T \mid X) = \prod_{s=1}^S C p(X_s \mid T) p(T) = \prod_{s=1}^S p(T \mid X_s), \quad (52) \]

meaning that the probability distribution of the multiple trajectories is the product of the probability distribution of all trajectories $p(T \mid X_s)$, defined by equation (15).

If we consider the covariance matrices $\Sigma$ are equal for every trajectories, the calculation is similar to the steps of (18) to (55), where we have

\[ \Phi_X = [\Phi_{X1}, \Phi_{X2}, \cdots, \Phi_{XS}], \quad \Phi_X \in \mathbb{R}^{N \times (S \sum_{s=1}^S L_s)}, \]

\[ v_X = [v_{X1}, v_{X2}, \cdots, v_{XS}], \quad v_X \in \mathbb{R}^{D \times (S \sum_{s=1}^S L_s)}, \quad (53) \]

which means that the size of $\Phi_X$ and $v_X$ depend on the size of each of the $S$ trajectories.

Therefore we have

\[ \log p(X \mid T) = \sum_{s=1}^S \log p(X_s \mid T) \]

\[ = \sum_{s=1}^S \log p(x_0^s) - \sum_{s=1}^S \sum_{t=1}^{L_s} \frac{1}{2} \log(2\pi)^D |\Sigma| \]

\[ - \sum_{s=1}^S \frac{1}{2} \|v_{X_s} - T \Phi_{X_s}\|_{\Sigma^{-1}}^2, \quad (54) \]
that leads to
\[
\log p(\mathcal{X} \mid T) = \sum_{s=1}^{S} \log p(x_0^s) - \sum_{s=1}^{S} \frac{L_s}{2} \log(2\pi)^D \mid \Sigma \mid - \frac{1}{2} \|v_{\mathcal{X}} - T\Phi_{\mathcal{X}}\|^2_{\Sigma^{-1}}. \tag{55}
\]

The prior can be calculated in a similar way with
\[
\log p(T) = \sum_{s=1}^{S} \log p(T) = \xi - \sum_{s=1}^{S} \frac{1}{2\rho} \|T^T\|^2_{\Lambda} = \xi - \frac{S}{2\rho} \|T^T\|^2_{\Lambda}. \tag{56}
\]

At last, the partial derivative is similar to (24) and is given by
\[
\frac{\partial}{\partial T} \log p(T \mid \mathcal{X}) = \frac{\partial}{\partial T} \log p(\mathcal{X} \mid T) + \frac{\partial}{\partial T} \log p_x(T)
\]
\[
= \Phi_{\mathcal{X}}v_{\mathcal{X}}^T\Sigma^{-1} - \Phi_{\mathcal{X}} \Phi_{\mathcal{X}}^T T^T \Sigma^{-1} - \frac{S}{\rho} \Lambda T^T, \tag{57}
\]

and the methods to compute \(T\) and the results will be similar to the single trajectory case.

The solution adopted is
\[
T = v_{\mathcal{X}} \Phi_{\mathcal{X}}^T \left( \Phi_{\mathcal{X}} \Phi_{\mathcal{X}}^T + \frac{S}{\rho} \Lambda \right)^{-1}. \tag{58}
\]

Note that
\[
\sum_{s=1}^{S} \left( \Phi_{\mathcal{X}} v_{\mathcal{X}}^s - \Phi_{\mathcal{X}} \Phi_{\mathcal{X}}^T T^s \right) \Sigma^{-1} - \frac{S}{\rho} \Lambda T^s = 0
\]
\[
\sum_{s=1}^{S} \left( \Phi_{\mathcal{X}} v_{\mathcal{X}}^s - \Phi_{\mathcal{X}} \Phi_{\mathcal{X}}^T T^s \right) \Sigma^{-1} - \frac{1}{\rho} \Lambda T^s = 0 . \tag{59}
\]

If we decide to use a simple prior \(p_x(T) = p(T)\), the vector field estimate yields
\[
T = v_{\mathcal{X}} \Phi_{\mathcal{X}}^T \left( \Phi_{\mathcal{X}} \Phi_{\mathcal{X}}^T + \frac{\sigma}{\rho} \Lambda \right)^{-1}, \tag{60}
\]

where it may forget the prior when \(S \to \infty\).
4.1 Recursive Vector Field estimation for Multiple trajectories

In the case of recursive vector fields for multiple trajectories, the method is also similar, but a special attention is needed for the priors. Here we use \( X_t = \{ X_1, \ldots, X_{S_t} \} \), where \( S_t \) is the number of trajectories which are variable at each time instant \( t \). For that reason, at each instant \( t \), we will have different matrix sizes for \( \Phi_{X_t} \) and \( \nu_{X_t} \).

The simplest approach is to follow the equations in section 3.1.1 and use \( p_X(T) = p(T) \). However, this approach tends to forget the prior as the number of trajectories grow. Nevertheless, we have

\[
T(t) = B(t)A(t)^{-1}.
\]

where, as in equation (32), we have

\[
B(t) = v_{X_t} \Phi_{X_t}^T + B(t - 1), \quad B(t - 1) = \sum_{i=1}^{t-1} \lambda^{t-1-i} v_{X_i} \Phi_{X_i}^T,
\]

\[
A(t) = \Phi_{X_t} \Phi_{X_t}^T + A(t - 1), \quad A(t - 1) = \sum_{i=1}^{t-1} \lambda^{t-1-i} \Phi_{X_i} \Phi_{X_i}^T,
\]

with \( A(0) = \frac{\sigma}{n} \Lambda \) and \( B(0) = 0 \).

It is possible have a recursive algorithm for this problem as follows

\[
K_t = \left[ I_{S_t} + \Phi_{X_t} P_{t-1} \Phi_{X_t} \right]^{-1} \Phi_{X_t} P_{t-1},
\]

\[
T_t = T_{t-1} + \left[ v_{X_t} - T_{t-1} \Phi_{X_t} \right] K_t,
\]

\[
P_t = P_{t-1} - P_{t-1} \Phi_{X_t} K_t.
\]

where all matrices except \( T \) and \( P \), have sizes dependable on the current number of trajectories \( S_t \) available.

Note that in this method, we may need some trajectory correction methods, so that one trajectory is not split into two trajectories by mistake. Moreover, an open problem is that it depends on the number of trajectories, which goes to infinity. Another problem is that the addiction of a new trajectory influences the entire vector field (through the prior \( \Lambda \)) even if it is a real small trajectory (outlier). A function that adjusts the area of influence of the prior given the position of a trajectory is open to be created.

4.2 Examples for multiple trajectories estimation

Figure 10 presents the result of a simulation using multiple identical trajectories in different time instants. The simple prior \( p_X(T) = p(T) \) was used. Figure 11 shows the difference of using the simple prior, or the first prior proposed in section 4. The result in figure 11b is similar to the result of a single trajectory.
Figure 10: Recursive vector field estimation for different time instants $t_1 = 200$, $t_2 = 500$, $t_3 = 700$ and $t_4 = 1099$ for multiple trajectories.

Figure 11: Recursive vector field estimation using: (a) simple prior, (b) the prior proposed in section 4.
\section{Multiple Vector Fields with Transition Matrix}

For multiple vector fields, we need an additional equation for the model

\begin{equation}
 p(k_t = j | k_{t-1} = i) = B_{ij},
\end{equation}

\begin{equation}
 x_t = x_{t-1} + T_{k_t}(x_{t-1}) + w_{k_t},
\end{equation}

where $T_{k_t}(x_{t-1})$ represents the velocity vector of the active vector field $k_t$ for the position $x_{t-1}$. Each element of the transition matrix $B_{ij}$ represents the switching probability from field $i$ to field $j$. In this section, $B_{ij}$ is fixed. The second equation in (64) is the same as (1), but here we have $w_{k_t} \sim \mathcal{N}(0, \Sigma_{k_t})$.

The conditional probability of $x_t$ is given by

\begin{equation}
 p(x_t | x_{t-1}, k_t) = \mathcal{N}_D(x_t | x_{t-1} + T_{k_t}(x_{t-1}), \Sigma_{k_t}),
\end{equation}

and

\begin{equation}
 \mathcal{N}_D(x_t | x_{t-1} + T_{k_t}(x_{t-1}), \Sigma_{k_t}) = \frac{1}{\sqrt{(2\pi)^D | \Sigma_{k_t}|}} e^{-\frac{1}{2}\|x_t - x_{t-1} - T_{k_t}(x_{t-1})\|^2 |_{\Sigma_{k_t}^{-1}}},
\end{equation}

We assume that $X_L = [x_0, x_1, \ldots, x_L] \in \mathbb{R}^{D \times (L+1)}$ is a matrix containing the positions of a single trajectory and $K_L = [k_0, \ldots, k_L] \in \{1, \ldots, K\}^{(L+1)}$ is the sequence of active vector fields, where $K$ is the number of possible vector fields. The joint probability of the complete trajectory is

\begin{equation}
 p(X_L, K_L) = p(x_0, x_1, \ldots, x_L, k_0, k_1, \ldots, k_L) = p(x_0, k_0) p(x_1, k_1 | x_0, k_0) \ldots p(x_L, k_L | x_{L-1}, k_{L-1}) = p(x_0, k_0) \prod_{t=1}^L p(x_t, k_t | x_{t-1}, k_{t-1}) = p(x_0, k_0) \prod_{t=1}^L p(x_t | x_{t-1}, k_t) p(k_t | x_{t-1}, k_{t-1}) = p(x_0, k_0) \prod_{t=1}^L \mathcal{N}_D(x_t | x_{t-1} + T_{k_t}(x_{t-1}), \Sigma_{k_t}) B_{k_{t-1}, k_t}.
\end{equation}

We can again perform space discretization and interpolate with

\begin{equation}
 T_{k_t}(x_{t-1}) = T_{k_t} \Phi_{x_t},
\end{equation}

and with $v_{x_t} \triangleq x_t - x_{t-1}$ and $\Phi_{x_t}$ as defined in equation (9). The multivariate gaussian distribution can be rewritten as

\begin{equation}
 p(x_t | x_{t-1}, k_t) = \frac{1}{\sqrt{(2\pi)^D | \Sigma_{k_t}|}} e^{-\frac{1}{2}\|v_{x_t} - T_{k_t} \Phi_{x_t}\|^2 |_{\Sigma_{k_t}^{-1}}}. 
\end{equation}
5.1 Multiple Vector Field Offline Estimation

Our objective is to estimate the parameter set $\mathcal{T} = \{T_1, \ldots, T_K\}$ given a trajectory $X$, as we did in section 2.3. We want to maximize

$$\hat{T} = \arg \max_T p(\mathcal{T} | X) = \arg \max_T p(X | \mathcal{T}) p(\mathcal{T})$$

$$= \arg \max_T \left( \log p(X | \mathcal{T}) + \log p(\mathcal{T}) \right) \quad (70)$$

to estimate $\mathcal{T}$, but the parcel containing $p(X | \mathcal{T})$ is unfeasible. However, we can use the Expectation-Maximization (EM) algorithm to estimate $\mathcal{T}$. We consider the complete joint distribution logarithm as

$$\log p(X, K, \mathcal{T}) = \log p(K | X, \mathcal{T}) + \log p(X | \mathcal{T}) + \log p(\mathcal{T}). \quad (71)$$

If we have an initial guess $\hat{T}$ and calculate the expected value $E\{\cdot | X, \hat{T}\}$ of equation (71) with respect to $p(K | X, \hat{T})$, we obtain

$$E\{\log p(X, K, \mathcal{T}) | X, \hat{T}\} = E\{\log p(K | X, \mathcal{T}) + \log p(X | \mathcal{T}) + \log p(\mathcal{T}) | X, \hat{T}\}, \quad (72)$$

which results on

$$U(\mathcal{T}, \hat{T}) = V(\mathcal{T}, \hat{T}) + \log p(X | \mathcal{T}) + \log p(\mathcal{T})$$

$$\Leftrightarrow U(\mathcal{T}, \hat{T}) - V(\mathcal{T}, \hat{T}) = \log p(X | \mathcal{T}) + \log p(\mathcal{T}). \quad (73)$$

Therefore, the expression

$$\arg \max_T p(\mathcal{T} | X) = \arg \max_T \left( \log p(X | \mathcal{T}) + \log p(\mathcal{T}) \right)$$

$$= \arg \max_T \left( U(\mathcal{T}, \hat{T}) - V(\mathcal{T}, \hat{T}) \right), \quad (74)$$

is true. We know that $V(\mathcal{T}, \hat{T}) \leq V(\hat{T}, \hat{T})$ and when we increase the value of $U(\mathcal{T}, \hat{T})$, we are also increasing the difference $U(\mathcal{T}, \hat{T}) - V(\mathcal{T}, \hat{T})$. Therefore we only need to maximize $U(\mathcal{T}, \hat{T})$.

The EM method is composed by the expectation (E-Step), where we compute the weights of each data sample corresponding to each vector field and the maximization (M-Step), where we estimate the vector fields based on the data and computed weights.

5.1.1 E-Step of EM

In this step, we compute the information needed to the M-Step. To know exactly what we need to compute, we disjoint $U(\mathcal{T}, \hat{T})$ into simpler parcels, starting with

$$U(\mathcal{T}, \hat{T}) = E\{\log p(X, K, \mathcal{T}) | X, \hat{T}\}$$

$$= E\{\log p(X, K | \mathcal{T}) | X, \hat{T}\} + \log p(\mathcal{T}) \ . \quad (75)$$
In the first parcel can use the expression from equation (67) and (69) to obtain

\[
E\{\log p(X, K \mid T) \mid X, \hat{T}\} = \sum_K p(K \mid X, \hat{T}) \log p(X, K \mid T) \\
= \sum_K p(K \mid X, \hat{T}) \log p(x_0, k_0) \\
+ \sum_K p(K \mid X, \hat{T}) \sum_{t=1}^L \log p(x_t \mid x_{t-1}, k_t) \\
+ \sum_K p(K \mid X, \hat{T}) \sum_{t=1}^L \log p(k_t \mid k_{t-1}),
\]

(76)

where only the second parcel depends on \( T \) and then we can rewrite equation (76) as

\[
E\{\log p(X, K \mid T) \mid X, \hat{T}\} = C + \sum_K p(K \mid X, \hat{T}) \left[ -\frac{1}{2} \sum_{t=1}^L \| v_{x_t} - T_{k_t} \Phi_{x_t} \|_{\Sigma_{k_t}}^2 \right],
\]

(77)

where \( C \) contains all the \( T \) independent parcels.

If we compute \( E\{f(k_t) \mid X, \hat{T}\} \) with respect to \( p(K \mid X, \hat{T}) \) and if the function \( f(k_t) \) depends only on \( k_t \), then we can discard all \( K \setminus \{k_t\} \)\(^5\) solutions, to have

\[
E\{f(k_t) \mid X, \hat{T}\} = \sum_K p(K \mid X, \hat{T}) f(k_t) \\
= \sum_{k_1, \ldots, k_L} p(k_1, \ldots, k_{t-1}, k_t, k_{t+1}, \ldots, k_L \mid X, \hat{T}) f(k_t) \\
= \sum_{K \setminus \{k_t\}, k_t} p(K \setminus \{k_t\}, k_t \mid X, \hat{T}) f(k_t) \\
= \sum_{k_t} \sum_{K \setminus \{k_t\}} p(K \setminus \{k_t\} \mid k_t, X, \hat{T}) p(k_t \mid X, \hat{T}) f(k_t) \\
= \sum_j^{K} p(k_t = j \mid X, \hat{T}) f(k_t = j) = \sum_j^{K} w_j(t) f(j),
\]

(78)

where \( w_j(t) \triangleq p(k_t = j \mid X, \hat{T}) \) is a weight that balance the samples of each vector field.

---

\(^5\)The expression \( K \setminus \{k_t\} = \{k_0, \ldots, k_{t-1}, k_{t+1}, \ldots, k_L\} = k^c_t \) means the complement of the set \( k_t \).
Therefore we can write equation (77) as

\[
E \{ \log p(X, K | T) | X, \hat{T} \} = C - \frac{1}{2} \sum_{t=L}^{T} \sum_{j=1}^{K} p(k_t = j | X, \hat{T}) \left[ v_{x_t} - T_j \Phi_{x_t} \|_{\Sigma_j}^2 \right] = C - \frac{1}{2} \sum_{t=1}^{T} \sum_{j=1}^{K} w_j(t) \| v_{x_t} - T_j \Phi_{x_t} \|_{\Sigma_j}^2 .
\] (79)

Each weight \( w_j(t) \) yields

\[
w_j(t) = p(k_t = j | X, \hat{T}) = \frac{\alpha_j(t)}{p(X | \hat{T})} \frac{p(x_0, \ldots, x_t, k_t = j | \hat{T})}{p(x_t | \hat{T})} = \frac{\alpha_j(t) \beta_j(t)}{\sum_{k=1}^{K} p(X, k_t = k | \hat{T})} = \frac{\alpha_j(t) \beta_j(t)}{\sum_{k=1}^{K} \alpha_k(t) \beta_k(t)},
\]

which can be computed using a forward-backward algorithm. The forward part of the algorithm yields

\[
\alpha_j(t) = p(x_0, \ldots, x_t-1, x_t, k_t = j | \hat{T}) = p(x_t | x_0, \ldots, x_t-1, k_t = j, \hat{T}) p(x_0, \ldots, x_t-1, k_t = j | \hat{T}) = p(x_t | x_{t-1}, k_t = j, \hat{T}) \sum_{i=1}^{K} p(x_0, \ldots, x_{t-1}, k_t = i, k_t = j | \hat{T}) \times p(x_0, \ldots, x_t-1, k_t = 1 | \hat{T}) = N_{D}(x_t | x_{t-1} + \hat{T}_j(x_{t-1}), \Sigma_j) \sum_{i=1}^{K} B_{ij} \alpha_i(t-1),
\]

with \( \alpha_i(0) = \pi_i = 1/K \), which satisfies \( \sum_{i=1}^{K} \pi_i = 1 \), for \( t = 1 \).

If the data concerning \( x_{t+1:L} \) is available, the backward part of the algo-
rithm yields
\[
\beta_j(t) = p(x_{t+1}, \ldots, x_L | x_0, \ldots, x_t, k_t = j, \hat{T}) = p(x_{t+1}, x_{t+2}, \ldots, x_L | x_t, k_t = j, \hat{T})
\]
\[
= \sum_{h=1}^{K} p(x_{t+1}, x_{t+2}, \ldots, x_L, k_{t+1} = h | x_t, k_t = j, \hat{T})
\]
\[
= \sum_{h=1}^{K} p(x_{t+1}, x_{t+2}, \ldots, x_L | x_t, k_t = j, k_{t+1} = h, \hat{T}) p(k_{t+1} = h | x_t, k_t = j, \hat{T})
\]
\[
= \sum_{h=1}^{K} \beta_h(t+1) N_D(x_{t+1} | x_t + \hat{T}_h(x_t), \Sigma_h) B_{jh},
\]
(82)

with \( \beta_j(L) = 1/K \), since \( x_{L+1} \) is not available.

If we use
\[
D_t = \text{diag} \begin{pmatrix} p(x_t | x_{t-1}, k_t = 1, \hat{T}) \\ \vdots \\ p(x_t | x_{t-1}, k_t = K, \hat{T}) \end{pmatrix},
\]
(83)
and
\[
B = \begin{bmatrix} B_{11} & \cdots & B_{1j} & \cdots & B_{1K} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{i1} & \cdots & B_{ij} & \cdots & B_{iK} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ B_{K1} & \cdots & B_{Kj} & \cdots & B_{KK} \end{bmatrix},
\]
(84)
then we can compute \( \alpha(t) = [\alpha_1(t), \ldots, \alpha_K(t)]^T \) and \( \alpha(t) = [\beta_1(t), \ldots, \beta_K(t)]^T \) using
\[
\alpha(t) = D_t B^T \alpha(t-1),
\]
\[
\beta(t) = B D_{t+1} \beta(t+1).
\]
(85)

The log \( p(T) \) parcel is the vector fields prior, similar to (21), but for multiple vector fields we have
\[
\log p(T) = \xi - \sum_{j=1}^{K} \frac{1}{2 \rho_j} \| T_j^T \|^2_\Lambda.
\]
(86)

From the E-Step, we only need to compute \( \alpha_j(t) \), \( \beta_j(t) \) and \( w_j(t) \) for the M-Step.
5.1.2 M-Step of EM

The maximization step consists in the differentiation of $U(T, \hat{T})$ with respect to each element $T_j$ in $T$

$$\frac{\partial}{\partial T_j} U(T, \hat{T}) \bigg|_{\hat{T}_j = T_j} = 0. \quad (87)$$

which can be obtained by differentiating equations (86) and (79) separately. Due to every vector fields $T_j$ does not depend on $T_{\gamma\neq j}$, we can use

$$\frac{\partial}{\partial T_j} \sum_{j=1, \ldots, \gamma, \ldots, K} f(T_j) = \frac{\partial}{\partial T_j} f(T_\gamma) \bigg|_{\gamma = j}, \quad (88)$$

which is the same as maximizing each vector field, independently.

The derivative of the prior from equation (86) yields

$$\frac{\partial}{\partial T_\gamma} \log p(T) = \frac{\partial}{\partial T_\gamma} \left[ -\sum_{j=1}^{K} \frac{1}{2\rho_j} \| T_j \|^2 \Lambda \right] = -\frac{1}{\rho_\gamma} \Lambda T_\gamma^T, \quad (89)$$

and the derivative of equation (79) yields

$$\frac{\partial}{\partial T_\gamma} E\{ \log p(X, K|T)|X, \hat{T} \} = \frac{\partial}{\partial T_\gamma} \left[ -\frac{1}{2} \sum_{t=1}^{L} \sum_{j=1}^{K} w_j(t) \| v_{x_t} - T_j \Phi_{x_t} \|_{\Sigma_{\gamma}^{-1}}^2 \right]$$

$$= \sum_{t=1}^{L} w_\gamma(t) \Phi_{x_t} (v_{x_t} - T_\gamma \Phi_{x_t})^T \Sigma_{\gamma}^{-1}. \quad (90)$$

Finally, we combine (89) and (90) to obtain

$$\frac{\partial}{\partial T_\gamma} U(T, \hat{T}) = \sum_{t=1}^{L} w_\gamma(t) \Phi_{x_t} (v_{x_t} - T_\gamma \Phi_{x_t})^T \Sigma_{\gamma}^{-1} - \frac{1}{\rho_\gamma} \Lambda T_\gamma^T, \quad (91)$$

which transposed and equaling to zero yields

$$\sum_{t=1}^{L} \Sigma_{\gamma}^{-1} (v_{x_t} - T_\gamma \Phi_{x_t}) \Phi_{x_t}^T w_\gamma(t) - \frac{1}{\rho_\gamma} T_\gamma \Lambda = 0$$

$$\Leftrightarrow \sum_{t=1}^{L} \Sigma_{\gamma}^{-1} v_{x_t} \Phi_{x_t}^T w_\gamma(t) - \sum_{t=1}^{L} \Sigma_{\gamma}^{-1} T_\gamma \Phi_{x_t} \Phi_{x_t}^T w_\gamma(t) - \frac{1}{\rho_\gamma} T_\gamma \Lambda = 0. \quad (92)$$
If we assume $\Sigma_\gamma = \sigma_\gamma I_D$, then

$$\sum_{t=1}^{L} v_{x_t} \Phi_{x_t} \frac{w_\gamma(t)}{\sigma_\gamma} - \sum_{t=1}^{L} T_\gamma \Phi_{x_t} \frac{w_\gamma(t)}{\sigma_\gamma} - \frac{1}{\rho_\gamma} T_\gamma \Lambda = 0$$

$$\Leftrightarrow T_\gamma = \left( \sum_{t=1}^{L} v_{x_t} \Phi_{x_t} \frac{w_\gamma(t)}{\sigma_\gamma} \right) \left( \sum_{t=1}^{L} \Phi_{x_t} \Phi_{x_t}^T \frac{w_\gamma(t)}{\sigma_\gamma} + \frac{1}{\rho_\gamma} \Lambda \right)^{-1}$$

$$\Leftrightarrow T_\gamma = \left( \sum_{t=1}^{L} v_{x_t} \Phi_{x_t} \Phi_{x_t}^T \frac{w_\gamma(t)}{\sigma_\gamma} \right) \left( \sum_{t=1}^{L} \Phi_{x_t} \Phi_{x_t}^T \frac{w_\gamma(t)}{\sigma_\gamma} + \frac{\sigma_\gamma}{\rho_\gamma} \Lambda \right)^{-1}.$$

which concludes the M-Step.

### 5.1.3 Expectation-Maximization algorithm

The steps for the batch EM algorithm are

1. Initialization: Begin iteration count with $l = 1$ and choose different initial vector fields $T^l_\gamma$, initial covariance matrices $\Sigma^l_\gamma$ and an initial weight $\pi^l_\gamma$ (usually $\pi^l_\gamma = 1/K$).

2. Compute the E-Step for all $t$ and $\gamma$
   
   2.1. Compute $N_D(x_t|x_{t-1} + T^l_\gamma(x_{t-1}), \Sigma^l_\gamma)$ with equation (69),
   
   2.2. Compute $\alpha^l_\gamma(t)$ and $\beta^l_\gamma(t)$ with equations (81) and (82),
   
   2.3. Compute $w^l_\gamma(t)$ with equation (80),

3. Compute the M-Step for all $t$ and $\gamma$

   Compute $T^{l+1}_\gamma$ with equations (93) and use $w^{l+1}_\gamma(t)$.

4. Increment iteration $l \leftarrow l + 1$ and repeat from step 2 until convergence or $l > l_{\text{max}}$.

Between the step (3.) and (4.) of the algorithm, we can calculate the expected value of the log-likelihood function with equation (75), which can be used instead to stop the iteration process once the convergence is achieved, p.e, when $T^{l+1}_\gamma - T^{l}_\gamma$ are less than a threshold.

### 5.2 Multiple Vector Field Online Estimation

The implementation of the batch EM algorithm in an online framework implies computational limits due to the need of memory capacity and speed. Various approaches can already implement Online EM.

In this work, our first approach is to update and iterate only the data observed at instant $t$, assuming it does not differ too much from previous observation. We use the forward algorithm for the E-Step

$$\alpha^l_\gamma(t) = N_D(x_t|x_{t-1} + \bar{T}_\gamma(t)\Phi_{x_t}, \Sigma_\gamma) \sum_{i=1}^{K} B_{i\gamma} \alpha^l_i(t-1),$$

35
where \( l \) is the iteration index of the EM algorithm. Note that the previous values \( \alpha_i(t-1) \) does not depend on iteration \( l \), because it is not updated.

The weight of the E-Step yields

\[
\hat{w}_\gamma^l(t) = \frac{\alpha_i^l(t)}{\sum_{k=1}^K \alpha_k^l(t)} .
\] (95)

We can compute vector field estimate as in section 3.1.1, where we have the estimation using the least squares algorithm

\[
K^l_\gamma(t) = \left[ w_\gamma(t) \right]^{-1} + \Phi_{x_t} P_\gamma(t-1) \Phi_{x_t} \right]^{-1} \Phi_{x_t} P_\gamma(t-1),
\]

\[
\hat{T}^{l+1}_\gamma(t) = \hat{T}_\gamma(t-1) + (v_{x_t} - \hat{T}_\gamma(t-1) \Phi_{x_t}) K^l_\gamma(t),
\] (96)

where the previous values \( \hat{T}_\gamma(t-1) \) and \( P_\gamma(t-1) \) do not change in the EM iteration algorithm.

In the end of the EM iterations, we update the values

\[
\hat{T}_\gamma(t) = \hat{T}^{l+1}_\gamma(t),
\]

\[
P_\gamma(t) = P_\gamma(t-1) - P_\gamma(t-1) \Phi_{x_t} K^l_\gamma(t),
\] (97)

where we can see that the covariance matrices \( P_\gamma(t) \) can be updated outside the EM iteration loop, saving computational time.

Examples of recursive estimation using \( K = 2 \) vector fields for instants \( t = \{25, 75, 150, 398\} \) are shown in figure 12.

For best understand the each vector field, we present figure 13. The figure 13c shows the distributions of the weights of each data sample to both vector fields. Due to the method used, where we do not update any past information, the first 25 are equally distributed over the two vector fields.

Further research intends to use adapt other EM algorithms (possibly the SMEM algorithm) to this problem, in order to obtain better results. Moreover we can expand to the analyze of multiple trajectories.

6 Transition Probabilities estimation

For multiple vector fields, we can define de general model

\[
p(k_t = j| k_{t-1} = i) = B_{ij},
\]

\[
x_t = x_{t-1} + T_{k_t}(x_{t-1}) + w_{k_t},
\] (98)

where now we intend to estimate the vector fields \( \mathcal{T} = \{T_1, \ldots, T_K\} \) and the transition matrix \( \mathcal{B} = \{B_{ij}\} \), which we include in parameter \( \theta = \{\mathcal{T}, \mathcal{B}\} \).

Most equations are equal to the ones in section 5 with the exception of the E-step and M-step of EM algorithm. The prior is \( \log p(\theta) = \log p(\mathcal{T}) + \log p(\mathcal{B}) \), where \( p(\mathcal{B}) \propto 1 \).
For the auxiliary function $U(\theta, \hat{\theta})$ we need to compute $E\{ f(k_{t-1}, k_t) | X, \hat{\theta} \}$ with respect to $p(K | X, \hat{\theta})$ and if the function $f(k_{t-1}, k_t)$ depends on $k_{t-1}$ and
$k_t$, then we can discard all $K \setminus \{k_{t-1}, k_t\}$ to have

$$E\{f(k_{t-1}, k_t) | X, \hat{\theta}\} = \sum_{k_{t-1}, ..., k_t} p(K | X, \hat{\theta}) f(k_{t-1}, k_t)$$

$$= \sum_{k_{t-1}, ..., k_t} p(k_1, ..., k_t, k_{t-1}, ..., k_L | X, \hat{\theta}) f(k_{t-1}, k_t)$$

$$= \sum_{k_{t-1}, ..., k_t} p(K \setminus \{k_{t-1}, k_t\}, k_{t-1}, k_t | X, \hat{\theta}) f(k_{t-1}, k_t)$$

$$= \sum_{k_{t-1}, k_t} p(K \setminus \{k_{t-1}, k_t\}, k_{t-1}, k_t, X, \hat{\theta}) \times$$

$$\times p(k_{t-1}, k_t | X, \hat{\theta}) f(k_{t-1}, k_t)$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{K} p(k_{t-1} = i, k_t = j | X, \hat{\theta}) f(k_{t-1} = i, k_t = j)$$

$$= \sum_{i=1}^{K} \sum_{j=1}^{K} w_{ij}(t) f(i, j),$$

(99)

where $w_{ij}(t) \triangleq p(k_{t-1} = i, k_t = j | X, \hat{\theta})$.

Each weight $w_{ij}(t)$ yields

$$w_{ij}(t) = p(k_{t-1} = i, k_t = j | X, \hat{\theta}) = p(X, k_{t-1} = i, k_t = j | \hat{\theta}) / p(X | \hat{\theta})$$

$$= p(x_0, \ldots, x_L, k_{t-1} = i, k_t = j | \hat{\theta}) / p(X | \hat{\theta})$$

$$= p(x_0, \ldots, x_t, x_{t+1}, \ldots, x_L, k_{t-1} = i, k_t = j | \hat{\theta}) / p(X | \hat{\theta})$$

$$= p(x_0, \ldots, x_t, k_{t-1} = i, k_t = j | \hat{\theta}) \times$$

$$\times p(x_{t+1}, \ldots, x_L | x_0, \ldots, x_t, k_{t-1} = i, k_t = j, \hat{\theta}) / p(X | \hat{\theta})$$

$$= p(x_0, \ldots, x_{t-1}, k_{t-1} = i, k_t = j | \hat{\theta}) \times$$

$$\times p(x_t | x_0, \ldots, x_{t-1}, k_{t-1} = i, k_t = j, \hat{\theta}) \beta_j(t) / p(X | \hat{\theta})$$

$$= \alpha_i(t-1) \frac{p(k_{t-1} = i, \hat{\theta})}{p(k_{t-1} = i, \hat{\theta})} B_{ij}$$

$$= \alpha_i(t-1) B_{ij} N(x_t | x_{t-1} + \hat{T}_j \Phi x_{t-1}, \Sigma_j) \beta_j(t) / p(X | \hat{\theta}),$$

(100)

where $p(X | \hat{\theta}) = \sum_{i,j} p(X, k_{t-1} = i, k_t = j, \hat{\theta})$ is a normalization constant. If data concerning $\{x_{t+1}, \ldots, x_L\}$ is not available, we can exclude $\beta_j(t)$. 

The first parcel of $U(\theta, \hat{\theta})$ is given by

$$
E\{\log p(X, K| \theta) | X, \hat{\theta}\} = \sum_{K} p(K| X, \hat{\theta}) \log p(X, k_0)
$$
$$
\quad + \sum_{K} p(K| X, \hat{\theta}) \sum_{t=1}^{L} \log p(x_t | x_{t-1}, k_t)
$$
$$
\quad + \sum_{K} p(K| X, \hat{\theta}) \sum_{t=1}^{L} \log p(k_t | k_{t-1})
$$
$$
= Q_1 + Q_2 + Q_3.
$$

In the M-Step, the vector field estimate was already mentioned and follows equation (93). The problem of maximizing the probability matrix from equation (101), $Q_3$, can be presented as

$$
\text{arg max}_{B_{ij}} Q_3 = \text{arg max}_{B_{ij}} \sum_{K} p(K| X, \hat{\theta}) \sum_{t=1}^{L} \log p(k_t = j | k_{t-1} = i)
$$
$$
= \text{arg max}_{B_{ij}} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{t=1}^{L} w_{ij}(t) \log B_{ij}, \quad \text{s.t.} \quad \sum_{j=1}^{K} B_{ij} = 1,
$$

which is an optimization problem with constraints. In this case, since $B_{ij}$ is not space dependent, we can define $W_{ij} = \sum_{t=1}^{L} w_{ij}(t)$. The restriction is independent for each line $i$, so we can simplify the problem by excluding the summation over $i$ and substitute the constrain in the optimization problem

$$
\text{arg max}_{B_{\alpha j}} Q_3^\alpha = \text{arg max}_{B_{\alpha j}} \sum_{j=1}^{K} W_{\alpha j} \log B_{\alpha j}, \quad \text{s.t.} \quad \sum_{j=1}^{K} B_{\alpha j} = 1
$$
$$
= \text{arg max}_{B_{\alpha j}} \sum_{j=2}^{K} W_{\alpha j} \log B_{\alpha j} + W_{\alpha 1} \log \left(1 - \sum_{j=2}^{K} B_{\alpha j}\right),
$$

for every $\alpha = 1, \ldots, K$ line. When we differentiate (103) we get

$$
\frac{\partial Q_3^\alpha}{\partial B_{\alpha \beta}} = W_{\alpha \beta} B_{\alpha \beta}^{-1} - W_{\alpha 1} \left(1 - \sum_{j=2}^{K} B_{\alpha j}\right)^{-1}
$$

which equaling to zero leads to

$$
W_{\alpha \beta} \left(1 - \sum_{j=2}^{K} B_{\alpha j}\right) - W_{\alpha 1} B_{\alpha \beta} = 0
$$
$$
\Leftrightarrow \quad W_{\alpha \beta} \left(1 - \sum_{j\neq \beta}^{K} B_{\alpha j} - B_{\alpha \beta}\right) - W_{\alpha 1} B_{\alpha \beta} = 0
$$
$$
\Leftrightarrow \quad B_{\alpha \beta} \left(1 + \frac{W_{\alpha 1}}{W_{\alpha \beta}}\right) + \sum_{j=2, j\neq \beta}^{K} B_{\alpha j} = 1,
$$

39
that is a linear equation. The solution is
\[ B_{\alpha\beta} = \frac{W_{\alpha\beta}}{\sum_{j=1}^{K} W_{\alpha j}}, \] (106)
which means that each line of \( B \) is simply the normalization of each line of \( W \equiv [W_{ij}] \).

### 6.1 Transition Probabilities Online Estimation

In the online estimation, we define
\[ W_{ij}(t) \triangleq \sum_{l=1}^{t} w_{ij}(l) = w_{ij}(t) + W_{ij}(t-1), \quad W_{ij}(0) = 0, \] (107)
and the online solution is
\[ B_{\alpha\beta}(t) = \frac{W_{\alpha\beta}(t)}{\sum_{j=1}^{K} W_{\alpha j}(t)} = \frac{W_{\alpha\beta}(t)}{W_{\alpha K}(t)}, \] (108)
where
\[ W_{\alpha K}(t) = \sum_{l=1}^{t} \sum_{j=1}^{K} w_{\alpha j}(l) = \sum_{l=1}^{t} W_{\alpha K}(l) = \sum_{j=1}^{K} W_{\alpha j}(l), \quad W_{\alpha K}(0) = 0, \] (109)
However, this solution overflows for \( t \to \infty \). We can use again the least squares method. First, we have
\[
C_t = c_t + C_{t-1} = W_{\alpha\beta}(t), \quad D_t = d_t + D_{t-1} = W_{\alpha K}(t), \quad X_t = C_t D_t^{-1} = B_{\alpha\beta}(t), \] (110)
Then, we develop
\[
X_t = C_t D_t^{-1} = c_t + X_{t-1} D_t^{-1} = c_t + X_{t-1} (D_t - d_t) D_t^{-1} = c_t + (X_{t-1} D_t - X_{t-1} d_t) D_t^{-1} 
= c_t + X_{t-1} + (c_t - X_{t-1} d_t) P_t.
\]
and using
\[
[A + B]^{-1} = [I + A^{-1} B]^{-1} A^{-1}. \] (112)
we get

\[ P_t = D_t^{-1} = [D_{t-1} + d_t]^{-1} = [P_{t-1}^{-1} + d_t]^{-1} = [1 + P_{t-1}d_t]^{-1}P_{t-1}. \]  

(113)

Therefore, the recursive solution is

\[
B_{\alpha\beta}(t) = B_{\alpha\beta}(t-1) + \left( w_{\alpha\beta}(t) - B_{\alpha\beta}(t-1)w_{\alpha K}(t) \right) P_{\alpha,\beta}(t) \\
P_{\alpha,\beta}(t) = \left[ 1 + P_{\alpha,\beta}(t-1)w_{\alpha K}(t) \right]^{-1} P_{\alpha,\beta}(t-1)
\]

(114)

which can be implemented for the complete transition matrix \( B \) using the Schur product

\[
B(t) = B(t-1) + \left( w(t) - B(t-1) \circ w_{K}(t) \right) \circ P(t) \\
P(t) = \left[ 1 + P(t-1) \circ w_{K}(t) \right]^{-1} \circ P(t-1)
\]

(115)

6.2 Examples for Online estimation of Transition Matrix

Examples of recursive estimation of vector fields and transition matrix using \( K = 2 \) vector fields for instants \( t = \{25, 75, 150, 398\} \) are shown in figure 14.

For best understand the each vector field, we present figure 15. The figure 15c shows the distributions of the weights of each data sample to both vector fields and figure 15d is the evolution of the transition matrix after each sample update.

7 Conclusions and Future Work

The work presented show some progress in the online implementations of project. The online estimation of a single vector field is equivalent to the offline solution and therefore there is no degradation in the online implementation. The estimation of multiple vector field can not be implemented in a strait-forward way since it is not possible the update all past samples due to computational memory limitation. For online implementation of the estimation of multiple vector fields, we can adopt some extensions of the Online EM algorithm. We used a simple approach, where we process only the new data. The solution seems reasonable, but yet lacks on the evaluation of he initial samples.

The estimation of a single transition matrix can be implemented in a similar way as the vector fields estimation, but it does not seem to improve much the results. However, we verify that the transition matrix tends to be close to an identity matrix.

In future work, we intend to find a working solution for the online implementation of the complete problem by expanding the so far solutions.
Figure 14: Recursive estimation of multiple vector fields and transition matrix.

Then we can search for methods more faster and robust, that can be implemented in harder situations.

After a reliable implementation is found and a good training data is obtained, we can advance to the detection of abnormal behavior in real-time situations.
Figure 15: Result of recursive estimation of multiple vector fields (a) and (b), weight of each sample (c) and evolution of the transition matrix (d).

References


**Appendix**

**A Matrix**

If two matrices $A \in \mathbb{R}^{l \times m}$ and $B \in \mathbb{R}^{m \times n}$ we have

\[
(AB)^T = B^T A^T, \tag{116}
\]

\[
(AB)^{-1} = B^{-1} A^{-1}, \tag{117}
\]

\[
(AB)^{-T} = A^{-T} B^{-T}, \tag{118}
\]

For the sum of matrices, we can use $X = [x_1, x_2, \ldots, x_N]$ such that

\[
x_1x_1^T + x_2x_2^T + \ldots + x_Nx_N^T = [x_1, x_2, \ldots, x_N][x_1, x_2, \ldots, x_N]^T = XX^T, \tag{119}
\]

and the same goes for $Y = [x_1^T, x_2^T, \ldots, x_n^T]$

\[
x_1^Tx_1 + x_2^Tx_2 + \ldots + x_N^Tx_N = [x_1^T, x_2^T, \ldots, x_N^T][x_1, x_2, \ldots, x_N]^T = YY^T, \tag{120}
\]

where $x_n$ are column vectors, which improves performance and precision.
B Trace

Being \( A \in \mathbb{R}^{n \times n} \) a matrix such that
\[
A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix},
\]
(121)
the matrix trace is
\[
\text{Tr} (A) = a_{11} + a_{22} + \ldots + a_{nn}.
\]
(122)

For \( A, B, C, D \) and \( X \) matrices and \( \alpha \) and \( \beta \) scalars, the trace properties are
\[
\text{Tr} (\alpha A + \beta B) = \alpha \text{Tr} (A) + \beta \text{Tr} (B),
\]
(123)
\[
\text{Tr} (AB) = \text{Tr} (B^T A^T),
\]
(124)
\[
\text{Tr} (ABC) = \text{Tr} (CAB) = \text{Tr} (BCA),
\]
(125)
\[
\text{Tr} (AXB + CXD) = \text{Tr} ((BA + DC) X),
\]
(126)
\[
\text{Tr} \left( A_1^T X A_1 + \ldots + A_n^T X A_n \right) = \text{Tr} \left( (A_1 A_1^T + \ldots + A_n A_n^T) X \right),
\]
(127)
if \( A = [A_1, \ldots, A_n] \), then
\[
\text{Tr} \left( A_1^T X A_1 + \ldots + A_n^T X A_n \right) = \text{Tr} (A A^T X) = \text{Tr} (A^T X A),
\]
(128)
For \( \alpha \) scalar, then \( \text{Tr} (\alpha) = \alpha \) and \( \alpha \text{Tr} (A) = \text{Tr} (\alpha A) \)

C Schur / Hadamard Product

The Schur product represents an element-by-element multiplication, also known as element-wise or entrywise product. For matrices with the same dimensions \( A, B, A \circ B \in \mathbb{R}^{n \times m} \), the Schur product is defined by\(^6\)
\[
A \circ B = \begin{bmatrix}
a_{11} & \cdots & a_{1m} \\
\vdots & \ddots & \vdots \\
a_{n1} & \cdots & a_{nm}
\end{bmatrix} \circ \begin{bmatrix}
b_{11} & \cdots & b_{1m} \\
\vdots & \ddots & \vdots \\
b_{n1} & \cdots & b_{nm}
\end{bmatrix} \triangleq \begin{bmatrix}
a_{11}b_{11} & \cdots & a_{1m}b_{1m} \\
\vdots & \ddots & \vdots \\
a_{n1}b_{n1} & \cdots & a_{nm}b_{nm}
\end{bmatrix},
\]
(129)
and is valid for higher dimensions \( A, B \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m} \).

The properties of commutation, association, distribution and transposition are applicable for the Schur product
\[
A \circ B = B \circ A,
\]
(130)
\(^6\)In Matlab the Schur product corresponds to the \( \cdot \) operation.
\[ A \circ (B \circ C) = (A \circ B) \circ C , \quad (131) \]
\[ A \circ (B + C) = A \circ B + A \circ C , \quad (132) \]
\[ (A \circ B)^T = A^T \circ B^T , \quad (133) \]

Other properties of the Schur product, for \( A, B \in \mathbb{R}^{n \times m}, X \in \mathbb{R}^{n \times n}, \)
\( c \in \mathbb{R}^n \) and \( d \in \mathbb{R}^m \), are\(^7\)
\[ \text{diag}(c)A = A \circ c \mathbf{1}_{1 \times m} , \quad (134) \]
\[ A \text{ diag}(d) = \mathbf{1}_{n \times 1} d^T \circ A , \quad (135) \]
\[ \text{diag}(A^T B)^T = \sum_n (A \circ B) \circ n , \quad (136) \]
\[ \text{diag}(A^T X B)^T = \sum_n (A \circ X B) = \sum_n (X^T A \circ B) , \quad (137) \]
\[ \text{Tr} \left( A^T B \right) = \sum_m \text{diag}(A^T B) = \sum_m \sum_n (A \circ B) , \quad (138) \]
\[ \text{Tr} \left( A^T X B \right) = \sum_m \sum_n (A \circ X B) = \sum_m \sum_n (X^T A \circ B) . \quad (139) \]

D  Determinant

The properties of a determinant of a matrix are
\[ \det AB = \det BA = \det A \det B , \quad (140) \]
\[ \det \alpha A = \alpha^n \det A , \quad (141) \]
\[ \det A^{-1} = (\det A)^{-1} , \quad (142) \]

E  Norms

Some norms to take in consideration.

For \( x \) vector,
\[ \|x\|^2 = x^T x . \quad (143) \]

For \( X \) matrix
\[ \|X\|^2 = \text{Tr} \left( X^T X \right) , \quad (144) \]
known as the Frobenius norm.

The same way we have for \( X \) and \( A \) matrices,
\[ \|X\|_A^2 \triangleq \text{Tr} \left( X^T A X \right) . \quad (145) \]

\(^7\)Consider that matrix multiplication \((XB \text{ or } X^T A)\) has priority (is calculated first) than the Schur multiplication.
F Matrix Inversion

For $A$ nonsingular

$$ [A + BC]^{-1} = A^{-1} - A^{-1}B[I + CA^{-1}B]^{-1}CA^{-1}, \quad (146) $$

$$ [A + B]^{-1} = A^{-1}[I + BA^{-1}]^{-1} = [I + A^{-1}B]^{-1}A^{-1}. \quad (147) $$

G Derivatives

For a matrix and $x$ vector, we have

$$ \frac{\partial Ax}{\partial x} = A, \quad \frac{\partial Ax}{\partial x^T} = A^T, \quad (148) $$

known as Jacobian.

For $A$ and $X$ matrices we have\(^8\)

$$ \frac{\partial \|X\|^2}{\partial X} = \frac{\partial \text{Tr}(X^TX)}{\partial X} = 2X^T, \quad (149) $$

$$ \frac{\partial \text{Tr}(AX)}{\partial X} = \frac{\partial \text{Tr}(XA)}{\partial X} = A, \quad (150) $$

$$ \frac{\partial \|X\|^2}{\partial A} = \frac{\partial \text{Tr}(X^TAX)}{\partial X} = X^T(A + A^T), \quad (151) $$

$$ \frac{\partial \text{Tr}(AXB)}{\partial X} = BA, \quad \frac{\partial \text{Tr}(AX^TB)}{\partial X} = A^TB^T, \quad (152) $$

$$ \frac{\partial \text{Tr}(B^TX^TAXB)}{\partial X} = BB^TX^T(A + A^T), \quad (153) $$

$$ \frac{\partial \|C - XB\|^2}{\partial X} = \frac{\partial \text{Tr}((C - XB)^TA(C - XB))}{\partial X} $$

$$ = -B(C - XB)^T(A + A^T). \quad (155) $$

$$ \frac{\partial \text{Tr}(AX^{-1}B)}{\partial X} = -X^{-1}BAX^{-1}, \quad (156) $$

$$ \frac{\partial \det AXB}{\partial X} = B(AXB)^A A, \quad (157) $$

$$ \frac{\partial \det AX^{-1}B}{\partial X} = -X^{-1}B(AXB)^A AX^{-1}, \quad (158) $$

\(^8\)Derivatives can only be calculated for scalar or vector functions, therefore we use the matrix trace or determinant.
\[
\frac{\partial \log \det AXB}{\partial X} = B(AXB)^{-1}A, \quad (159)
\]
\[
\frac{\partial \log \det AX^{-1}B}{\partial X} = -X^{-1}B(AX^{-1}B)^{-1}AX^{-1}, \quad (160)
\]
\[
\frac{\partial \log \det X^{-1}}{\partial X} = -\frac{\partial \log \det X}{\partial X} = -X^{-1}, \quad (161)
\]

Moreover, if \( A = A^T \), then \( A + A^T = 2A \), and we can obtain
\[
\frac{\partial \|X\|_A^2}{\partial X} = \frac{\partial \text{Tr} \left( X^TAX \right)}{\partial X} = 2X^TA, \quad (162)
\]
\[
\frac{\partial \text{Tr} \left( B^T X^TAXB \right)}{\partial X} = 2BB^TX^TA, \quad (163)
\]
\[
\frac{\partial \|C - XB\|_A^2}{\partial X} = -2B(C - XB)^TA. \quad (164)
\]

### H Probabilities

For \( x \in \{x_1, x_2, \ldots, x_n\} \), we denote that \( p(x_1) \) is the probability of \( x = x_1 \), \( \text{i.e.}, p(x = x_1) \) and can be represented as
\[
p(x) \propto \begin{bmatrix} p(x_1) \\ p(x_2) \\ \vdots \\ p(x_n) \end{bmatrix}, \quad \text{where } \sum_x p(x) = 1 .
\]

For \( x \) and \( y \) the joint probability is given by \( p(x, y) \) and can be represented as
\[
p(x, y) \propto \begin{bmatrix} p(x_1, y_1) & p(x_1, y_2) & \cdots & p(x_1, y_n) \\ p(x_2, y_1) & p(x_2, y_2) & \cdots & p(x_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ p(x_n, y_1) & p(x_n, y_2) & \cdots & p(x_n, y_n) \end{bmatrix}, \quad \text{where } \sum_{x,y} p(x, y) = 1 .
\]

To get \( p(x) \) from \( p(x, y) \) we do
\[
p(x) = \sum_y p(x, y) .
\]

Conditional probability is defined by
\[
p(x | y) = \frac{p(x, y)}{p(y)} \quad \Leftrightarrow \quad p(x, y) = p(x | y)p(y) .
\]
If $x$ and $y$ are independent, we can do

$$p(x, y) = p(x)p(y), \quad \text{and} \quad p(x|y) = p(x).$$

The Bayes’ law tells that

$$p(x, y) = p(x|y)p(y) = p(y|x)p(x) \iff p(x|y) = \frac{p(y|x)p(x)}{p(y)}.$$  

If $x$ is conditionaly independente of $z$ given $y$, then

$$p(x|y, z) = p(x|y).$$

## 1 Sum of Norms

We know that

$$\|v_{x_t} - T\Phi_{x_t}\|_{\Sigma^{-1}}^2 = \text{Tr}\left( (v_{x_t} - T\Phi_{x_t})^T \Sigma^{-1} (v_{x_t} - T\Phi_{x_t}) \right)$$

$$= \text{Tr}\left( (v_{x_t} - T\Phi_{x_t}) (v_{x_t} - T\Phi_{x_t})^T \Sigma^{-1} \right)$$

$$= \text{Tr}\left( (v_{x_1}v_{x_t}^T - 2T\Phi_{x_t}v_{x_t}^T + T\Phi_{x_t}(\Phi_{x_t})^T) \Sigma^{-1} \right).$$

If we have a sum of norms, we shall have a sum of traces, i.e.

$$\sum_{t=1}^{L} \|v_{x_t} - T\Phi_{x_t}\|_{\Sigma^{-1}}^2 = \|v_{x_1} - T\Phi(x_1)\|_{\Sigma^{-1}}^2 + \ldots + \|v_{x_L} - T\Phi(x_L)\|_{\Sigma^{-1}}^2$$

$$= \text{Tr}\left( (v_{x_1}v_{x_1}^T - 2T\Phi(x_1)v_{x_1}^T + T\Phi(x_1)(\Phi(x_1))^T) \Sigma^{-1} \right)$$

$$\ldots$$

$$+ \text{Tr}\left( (v_{x_L}v_{x_L}^T - 2T\Phi(x_L)v_{x_L}^T + T\Phi(x_L)(\Phi(x_L))^T) \Sigma^{-1} \right).$$

we are able to group some parcels such that

$$\sum_{t=1}^{L} \|v_{x_t} - T\Phi_{x_t}\|_{\Sigma^{-1}}^2 = \text{Tr}\left( (v_{x_1}v_{x_1}^T + \ldots + v_{x_L}v_{x_L}^T)$$

$$- 2T(\Phi(x_1)v_{x_1}^T + \ldots + \Phi(x_L)v_{x_L}^T)$$

$$+ T(\Phi(x_1)(\Phi(x_1))^T + \ldots + \Phi(x_L)(\Phi(x_L))^T) \Sigma^{-1} \right).$$

We consider that

$$\Phi_X = [\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_L)]$$

$$v_X = [v_{x_1}, v_{x_2}, \ldots, v_{x_L}],$$

(165)
so we can get

\[ v_X^T v_X = v_{x_1}^T v_{x_1} + \ldots + v_{x_L}^T v_{x_L} \]

\[ \Phi_X^T v_X = \Phi(x_1) v_{x_1}^T + \ldots + \Phi(x_L) v_{x_L}^T \]

\[ \Phi_X \Phi_X^T = \Phi(x_1) \Phi(x_1)^T + \ldots + \Phi(x_L) \Phi(x_L)^T . \]

Therefore, we can get

\[
\sum_{t=1}^{L} \left\| v_{x_t} - T \Phi_{x_t} \right\|_{\Sigma^{-1}}^2 = \text{Tr} \left( \left( v_X^T v_X - 2T \Phi_X^T v_X + T(\Phi_X \Phi_X^T) T^\top \right) \Sigma^{-1} \right) \\
= \text{Tr} \left( \left( v_X - T \Phi_X \right)^T \Sigma^{-1} \left( v_X - T \Phi_X \right) \right) \\
= \left\| v_X - T \Phi_X \right\|_{\Sigma^{-1}}^2 .
\]

### J Gaussian Distribution

Being \( x \sim \mathcal{N}(\mu, \rho) \) with data \( x \in \mathbb{R} \), mean \( \mu \) and variance \( \rho \), then

\[ p(x) = \frac{1}{\sqrt{2\pi \rho}} e^{-\frac{(x-\mu)^2}{2\rho}} . \tag{166} \]

Being \( x \sim \mathcal{N}_D(\mu, \Sigma) \) with data \( x = [x_1, x_2, \ldots, x_D]^T \in \mathbb{R}^D \), mean \( \mu \) and covariance matrix \( \Sigma \), then

\[ p(x) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2} ||x-u||^2_{\Sigma^{-1}}} = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{-\frac{1}{2} (x-u)^T \Sigma^{-1} (x-u)} , \tag{167} \]

where \( |\Sigma| = \text{det}(\Sigma) \).
K  Obtain neighbor matrix

Considering the set of nodes in figure 16. We define that the neighbors of one node are the closest nodes in all vertical, horizontal and diagonal directions.

Figure 16: Node grid.

Therefore, to know which nodes have neighbors, we construct table 2 where 1 means that nodes are neighbors and 0 means no neighbors.

Table 2: Neighbor table.

<table>
<thead>
<tr>
<th></th>
<th>t_1</th>
<th>t_2</th>
<th>t_3</th>
<th>t_4</th>
<th>t_5</th>
<th>t_6</th>
<th>t_7</th>
<th>t_8</th>
<th>t_9</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t_5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t_6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t_7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t_8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t_9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Then, for each column we compute the difference operator matrix \( v \). An example is, considering the first two columns, we would get

\[
\begin{bmatrix}
1 & 1 \\
1 & 1 \\
0 & 1 \\
1 & 1 \\
0 & 1 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\Rightarrow v' =
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad (168)
\]
that makes each column to have only two nodes with different signals, which allows to operate the all difference between nodes separately. Being \( v \) the complete operator matrix, the restrictions concluded are that
\[
\sum_i v_{ij} = 0 \quad \text{and} \quad \sum_i |v_{ij}| = 2 ,
\] (169)

\section{Recursive least square}

When we have an equation as
\[
b_t = X_t a_t \iff b_t a_t^T = X_t a_t a_t^T \iff B_t = X_t A_t ,
\] (170)

where \( a_t \in \mathbb{R}^N \), \( b_t \in \mathbb{R}^D \) with \( S_t \) a variable dimension representing the number of trajectories at each instant \( t \). The matrices \( A_t \in \mathbb{R}^{N \times N} \) and \( B_t \in \mathbb{R}^{D \times N} \) are used to compute the solution \( X_t \in \mathbb{R}^{D \times N} \) as

When we have an equation as
\[
b_t = X_t a_t \iff b_t a_t^T = X_t a_t a_t^T \iff B_t = X_t A_t ,
\] (171)

then the solution of \( X_t \) is
\[
X_t = B_t A_t^{-1} ,
\] (172)

where recursively, we have
\[
A_t = \sum_{i=0}^{t} \lambda^{t-i} a_ia_i^T
= a_t a_t^T + \sum_{i=0}^{t-1} \lambda^{t-i} a_ia_i^T
\] (173)

and the same way we have
\[
B_t = \sum_{i=0}^{t} \lambda^{t-i} b_ia_i = b_t a_t^T + \lambda B_{t-1} .
\] (174)
Therefore, the recursive least square are achieved with

\[ X_t = B_t A_t^{-1} \]
\[ = [\lambda B_{t-1} + b_t a_t^T] A_t^{-1} \]
\[ = [X_{t-1} \lambda A_{t-1} + b_t a_t^T] A_t^{-1} \]
\[ = [X_{t-1} A_t - a_t a_t^T + b_t a_t^T] A_t^{-1} \]
\[ = [X_{t-1} A_t - X_{t-1} a_t a_t^T + b_t a_t^T] A_t^{-1} \]
\[ X_t = X_{t-1} + [b_t - X_{t-1} a_t] a_t^T A_t^{-1} \]
\[ X_t = X_{t-1} + [b_t - X_{t-1} a_t] a_t^T P_t . \]

By the fact that

\[ [A + BC]^{-1} = A^{-1} - A^{-1} B [I + CA^{-1} B]^{-1} CA^{-1} , \]

then

\[ P_t = A_t^{-1} = [\lambda A_{t-1} + a_t a_t^T]^{-1} \]
\[ = \frac{P_{t-1}}{\lambda} - \frac{P_{t-1} a_t}{\lambda^2} \left[ 1 + a_t^T P_{t-1} a_t \right]^{-1} a_t^T P_{t-1} a_t \]
\[ = \frac{P_{t-1}}{\lambda} - \frac{P_{t-1} a_t}{\lambda (\lambda + a_t^T P_{t-1} a_t)} \]
\[ = [P_{t-1} - P_{t-1} a_t K_t] / \lambda , \]

with

\[ K_t = \frac{a_t^T P_{t-1}}{\lambda + a_t^T P_{t-1} a_t} . \]

Moreover we have

\[ a_t^T P_t = a_t^T [P_{t-1} - P_{t-1} a_t a_t^T P_{t-1} a_t] / \lambda \]
\[ = [I - \frac{a_t^T P_{t-1} a_t}{\lambda + a_t^T P_{t-1} a_t}] a_t^T P_{t-1} a_t / \lambda \]
\[ = \frac{[\lambda + a_t^T P_{t-1} a_t - a_t^T P_{t-1} a_t] a_t^T P_{t-1} a_t}{\lambda + a_t^T P_{t-1} a_t} \]
\[ = \frac{a_t^T P_{t-1}}{\lambda + a_t^T P_{t-1} a_t} = K_t , \]

and the final form of the algorithm is given by

\[ K_t = \frac{a_t^T P_{t-1}}{\lambda + a_t^T P_{t-1} a_t} \]
\[ X_t = X_{t-1} + [b_t - X_{t-1} a_t] K_t \]
\[ P_t = [P_{t-1} - P_{t-1} a_t K_t] / \lambda . \]
M Recursive Least Squares for Multiple inputs and outputs

When we have an equation as

\[ b_t = X_t a_t \iff b_t a_t^T = X_t a_t a_t^T \iff B_t = X_t A_t, \]

where \( a_t \in \mathbb{R}^{N \times S_t} \), \( b_t \in \mathbb{R}^{D \times S_t} \) with \( S_t \) a variable dimension representing the number of trajectories at each instant \( t \). The matrices \( A_t \in \mathbb{R}^{N \times N} \) and \( B_t \in \mathbb{R}^{D \times N} \) are used to compute the solution \( X_t \in \mathbb{R}^{D \times N} \) as

\[ X_t = B_t A_t^{-1}, \]

where recursively, we have

\[
A_t = \frac{1}{\sigma} \sum_{i=0}^{t} \lambda^{t-i} a_i a_i^T
\]

\[
= \frac{1}{\sigma} a_t a_t^T + \frac{1}{\sigma} \sum_{i=0}^{t-1} \lambda^{t-i} a_i a_i^T
\]

\[
= \frac{1}{\sigma} a_t a_t^T + \lambda \sum_{i=0}^{t-1} \lambda^{t-1-i} a_i a_i^T
\]

\[
= \frac{1}{\sigma} a_t a_t^T + \lambda A_{t-1}^{-1} A_{t-1}^{-1}
\]

and the same way we have

\[
B_t = \frac{1}{\sigma} \sum_{i=0}^{t} \lambda^{t-i} b_i a_i = \frac{1}{\sigma} b_t a_t^T + \lambda B_{t-1}.
\]

Therefore, the recursive least square are achieved with

\[
X_t = B_t A_t^{-1}
\]

\[
= \left[ \frac{1}{\sigma} b_t a_t^T + \lambda B_{t-1} \right] A_t^{-1}
\]

\[
= \left[ \frac{1}{\sigma} b_t a_t^T + X_{t-1} \lambda A_{t-1}^{-1} \right] A_t^{-1}
\]

\[
= \left[ \frac{1}{\sigma} b_t a_t^T + X_{t-1} A_t - X_{t-1} \frac{1}{\sigma} a_t a_t^T \right] A_t^{-1}
\]

\[ X_t = X_{t-1} + (b_t - X_{t-1} a_t) a_t^T A_t^{-1} / \sigma \]

\[ X_t = X_{t-1} + (b_t - X_{t-1} a_t) a_t^T P_t / \sigma. \]

By the fact that

\[
[A + B C]^{-1} = A^{-1} - A^{-1} B [I + C A^{-1} B]^{-1} C A^{-1},
\]
then
\[ P_t = A_t^{-1} = \left[ \frac{1}{\sigma}a_t a_t^T + \lambda A_{t-1} \right]^{-1} \]
\[ = \frac{P_{t-1}}{\lambda} - \frac{P_{t-1}}{\lambda} \frac{1}{\sigma} \left[ I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} a_t^T P_{t-1}^{-1} \]
\[ = \left[ P_{t-1} - P_{t-1} a_t \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} a_t^T \right] / \lambda, \tag{187} \]

where \( I_{S_t} \) is the identity matrix with dimensions \( S_t \times S_t \).

We assume de Kalman gain
\[ K_t = \frac{1}{\sigma} a_t^T P_t = \frac{a_t^T}{\lambda} \left[ P_{t-1} - P_{t-1} a_t \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} a_t^T P_{t-1} \right] \]
\[ = \frac{1}{\lambda} \left[ a_t^T P_{t-1} - a_t^T P_{t-1} a_t \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} a_t^T P_{t-1} \right] \]
\[ = \frac{1}{\lambda} \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1} a_t \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} a_t P_{t-1} \right] \]
\[ = \frac{1}{\lambda} \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1} a_t \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1}^{-1} a_t \right]^{-1} \right] a_t P_{t-1} \]
\[ = \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1} a_t \right]^{-1} a_t P_{t-1} \tag{188} \]

The final form of the algorithm yields
\[ K_t = \left[ \lambda \sigma I_{S_t} + a_t^T P_{t-1} a_t \right]^{-1} a_t P_{t-1} \]
\[ X_t = X_{t-1} + [b_t - X_{t-1} a_t] K_t \]
\[ P_t = [P_{t-1} - P_{t-1} a_t K_t] / \lambda. \tag{189} \]

\section*{N Recursive Estimation with forgetting factor}

We have the covariance matrix
\[ P_t = A'(t)^{-1} = \left[ \frac{1}{\sigma} \Phi_{x_t} \Phi_{x_t}^T + \lambda A'(t-1) + \frac{1-\lambda}{\rho} A \right]^{-1} \]
\[ = \left[ \lambda P_{t-1}^{-1} + \frac{1}{\sigma} \Phi_{x_t} \Phi_{x_t}^T + \frac{1-\lambda}{\rho} A \right]^{-1}, \tag{190} \]

and using
\[ [A + B]^{-1} = A^{-1} [I + BA^{-1}]^{-1}. \tag{191} \]

results on
\[ P_t = \frac{P_{t-1}}{\lambda} \left[ I_N + \left( \frac{1}{\sigma} \Phi_{x_t} \Phi_{x_t}^T + \frac{1-\lambda}{\rho} A \right) P_{t-1} \right]^{-1} \]
\[ = P_{t-1} \left[ \lambda I_N + \left( \frac{1}{\sigma} \Phi_{x_t} \Phi_{x_t}^T + \frac{1-\lambda}{\rho} A \right) P_{t-1} \right]^{-1}, \tag{192} \]
The vector field is given by

\[ T_t = B(t)P_t = \left[ \frac{1}{\sigma} v_x \Phi X_t + \lambda B(t - 1) \right] P_t \]
\[ = \left[ \frac{1}{\sigma} v_x \Phi X_t + T_{t-1} \lambda A'(t - 1) \right] P_t \]
\[ = \left[ \frac{1}{\sigma} v_x \Phi X_t + T_{t-1} \left[ A'(t) - \frac{1}{\sigma} \Phi_x \Phi x_t - \frac{1-\Delta}{\rho} \Lambda \right] \right] P_t \]
\[ = T_{t-1} A'(t) P_t + \left[ \frac{1}{\sigma} v_x \Phi X_{t-1} - T_{t-1} \frac{1}{\sigma} \Phi_x \Phi x_{t-1} - T_{t-1} \frac{1-\Delta}{\rho} \Lambda \right] P_t \]
\[ = T_{t-1} + \left[ \frac{1}{\sigma} (v_x - T_{t-1} \Phi X_t) \Phi X_{t-1} - T_{t-1} \frac{1-\Delta}{\rho} \Lambda \right] P_t, \]