Practical exam of Cognitive Data Fusion

Jorge Leonardo Quimi Villon

24/04/2020

Introduction

The problem of estimating models from a reduced set of training data is relevant task that allows system to learn dynamic models. State estimation is important for motion prediction, tracking, identification of temporal patterns, autonomous vehicle and abnormality detection. We are interested in filter self-detecting abnormalities [1] of a moving agent from its trajectory data generated by a planned activity. The planned activity is a sequence of actions (or state transitions) that the agent performs to accomplish a task from a certain starting state, in a certain environment. The sequence of actions can be learned from training data grouped into discrete motion patterns. Abnormalities can be defined as observations that do not match with learned action patterns and models for abnormality detection are generally trained on a set of observations of normal activities.

Several state estimation methods have been employed for modeling linear and nonlinear dynamic. An example for Bayesian filter using a dynamic model is the Kalman Filter (KF), an optimal state estimator for linear dynamic system with Gaussian noise iteratively predicting and estimating states from incremental observations. The posterior probability density function (PDF) of the continuous state in the KF is Gaussian. A Particle Filter (PF) allows one to manage a numerical approximation of the posterior in a nonlinear, non-Gaussian Bayesian filtering problem. PF approximates $PDFs$ with a set of weighted samples (particles), which are propagated with an iterative random process. Particle are re-weighted based on dynamics and a likelihood score that depends on new observations.

In this report, we present a method to jointly learn coupled KF and PF models, which we call Markov Jump Particle Filter ($MJPF$), and that we use for multidimensional probabilistic anomaly measurement. As un-supervised learning method we use Self-organizing Maps (SOM) to learn switching variables vocabulary. This report proposes a strategy to incremental learning and abnormality detection task.

Fig. 3 shows the block diagram of the proposed method.

This report is structured as follows: Section 1 presents the training and testing dataset. Section 2 presents a discussion on the training phase. section 3 results of the testing phase and section 4 shows the conclusions and possible future developments.

1. Dataset

In this project, a dataset describing trajectories is given (in terms of positional data) of two entities that are interacting. This set of trajectories are provided through the simulation tool seen in **lab 3** (simulator). Two types of data are given, one for the training phase and one for the testing phase.

During the test phase, it may be expected that the entities interact according to the learned dynamic models, but it is not always the case. There are situations which involves some sort of abnormalities. An abnormal situation is any situation which is different from the observed situation during the training phase. In Fig. 2, you can see (in green dots) the trajectory of an obstacle, so follower needs to bend the normal path in order to avoid the obstacle. This scenario can be considered as an abnormal situation.

The goals of this project are to learn the two entities (follower and attractor) separately in the normal situation, then learned the interaction between these entities and, try to detect any deviated situation from the learned normality with and without interaction between these two entities.

1.1 Training dataset

In each scenario, one object pulls its direction towards the other object. The first object referred as a "follower" and the second one addressed as "attractor". In a normal (training phase) situation (Fig. 1) follower with different starting point directly moves towards the attractor. Where it is not influenced by any other forces but the attractor object. In Fig. 1 you can observe the trajectory of attractor the red dots. This shows that the attractors is moving in this case. This phase is referred as "training" task.

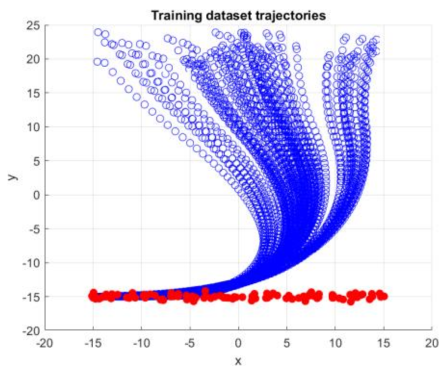


Fig. 1 *Trajectories of follower and attractor*

1.2 Testing dataset

In abnormal (testing phase) situation (Fig. 2Fig. 1) follower with different starting point directly move towards the attractor by avoiding an obstacle in environment, in this case the obstacle is modeled as a moving trajectory (in green) that the followers will have to avoid. Trajectory in red color shows the presence of moving attractor in the environment. This shows that the attractor is moving in this case. This phase referred as "testing" phase.

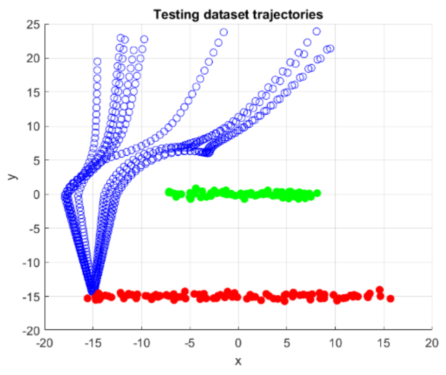


Fig. 2 *Trajectories of follower (blue circle), attractor (red dots) and obstacle (green dots)*

2. Training phase

Fig. 3 shows the procedure of incremental learning from errors. First you will take the errors (abnormalities) from the Null Force Filter for training dataset and these abnormalities you will learn new GDBN model as shown in Fig. 3.

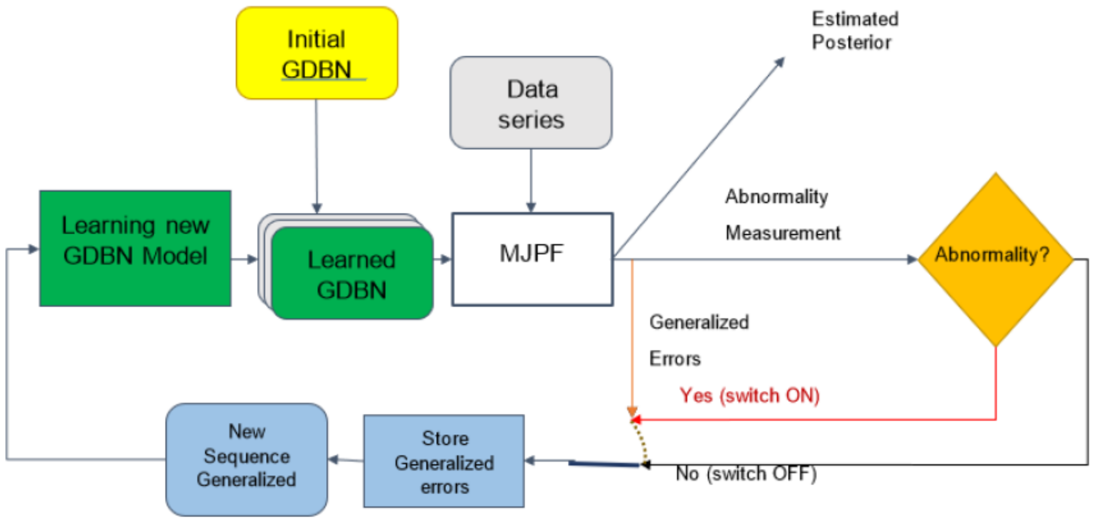


Fig. 3 *Block Diagram for the incremental Learning of Bayesian Filter*.

Data series (grey block in Fig. 3) consists of two types of dataset, training and testing. First start with the training dataset. The implementation of these blocks (grey block in Fig. 3) are explained in detail in later section.

2.1 Null Force Filter

According to the general scheme of a self-improving estimation filter, the current set of available (i.e. previously learned) models is tested on a sequence of samples with the objective to verify its prediction capability.

Once again, this measure of accuracy is conveyed by the innovation vector and is employed to identify when the filter being considered is no more appropriate to describe the agent.

In other words, high values of the innovation (i.e. upon a certain threshold) describe the presence of abnormalities since they appear whenever none of the known models provides an appropriate inference and therefore whenever the behavior of the agent deviates from any of the feasible (i.e. expected) modalities.

In these situations, the errors can be thought as descriptors of the unknown dynamics that have generated them and therefore employed (through clusterizatio) to extract these modalities allowing the MJPF to increase its overall knowledge about the agent Fig. 3.
The computation of the prediction errors (i.e. innovations) is possible only supposing to have already fixed some initial model through which the inference can actually be generated.
This means that, at the beginning, the MJPF has to be initialized with some pre-defined (i.e. not derived from the observation of the agent) dynamics.
As it can be easily understood from the analysis of the “UMKF” script, in the considered case, the first set of innovations is derived from a Kalman filter working under the null-force assumption (i.e. UMKF).
After a first phase, which coincides with the elimination of all the zero values that were synthetically introduced in the training trajectories to make all of them uniform in terms of dimension, the structure of the pre-defined filter is described.
Coherently with the null-force hypothesis, the linear dynamic model (i.e. $x_{k+1} = A * x_k + w$) employed for the generation of inferences is described by an identity matrix (i.e. $A = I$) thus specifying that, a part from the uncertainty, the prediction of the next state (i.e x_{k+1}) will coincide with the last updated estimate (i.e. x_k).
In other words, the supposition about the next evolution of the agent is based on the idea that both position and velocity will not change from stage k to stage $k + 1$.
From the “mechanical” point of view, the attractor (i.e. the point with the minimum potential) is always supposed to be in the current location thus making the force and the acceleration acting on the agent equal to zero and keeping the velocity always equal to itself.
In particular, the initial value of the velocity (at each prediction step) is fixed equal to zero and, for what has been said before, it will remain so for all the iterations.
By considering again the “mechanical” interpretation, this situation coincides with assuming that, in any moment in which the agent is considered, its kinetic energy is null and therefore the only possible movement is due to the attraction (i.e. the movement is generated due to the descending of the bell-shaped potential energy).
With the null velocity, of course, even the position is assumed by the filter to be time invariant.
Under these conditions, a “static” inference will be generated at each stage by the “kf_predict” function.

The comparison of this value with the observation is performed in the “kf_update” function and allows both to gather the innovation describing how much the behavior of the agent disagrees with the expected one and to update the prediction generating the final estimate for the current stage.

Assuming to know the measurement model, it is possible to compute an expected measure as $H * x_k$ with x_k representing the inference obtained as a projection, through the chosen (i.e. static) dynamic model, according to the expression $x_k = A * x_{k-1} = I * x_{k-1} = x_{k-1}$. As the H matrix simply extracts the first two components of the state vector x_k, the expected measurement $H * x_k = H * x_{k-1}$ coincides with the position of the agent at the previous stage.

Since the innovation is computed as the difference between the true and the predicted observations (i.e. $inn = z_k - H * x_{k-1}$), it will coincide with the difference between two position vectors at stages $k-1$ and k and therefore it can be interpreted as a velocity estimate.

This allows the generation of a matrix MM containing, for each sample of each training trajectory, an overall estimation vector of both the position and velocity of the agent for the current iteration.

Finally, this information, together with other quantities is stored in the output structure “dataFollower” and “dataAttractore”, which will be used in the following clustering phase.

In Fig. 1 all the training trajectories are scattered. On each of them the filter will apply its estimation process with the objective to generate a sequence of innovations. These innovations represent abnormalities with respect to the null-force (i.e. static) hypothesis and therefore will be employed to extract the set of unknown dynamics embedded in the trajectories themselves.

In Fig. 4 , four different subplots are displayed. In the first, the observations regarding a single one (randomly chosen) of the training trajectories are represented.

In the second plot, the result of the estimation process is shown. According to the standard scheme of a Kalman filter, the scattered points, which represent the filtered trajectory (i.e. the position estimates), are obtained, at each iteration, by modifying (i.e. updating) a prediction (in this case based on the null-force hypothesis), through the evidence carried by the observation.

On the other hand, the red arrows represent the innovation, which describes a difference in the position vectors suggesting the presence of a non-null motion (i.e. velocity).

In the third plot, the filtered trajectory is again represented but, differently from the previous case, a noisy version of the innovation vectors is plotted. This version considers the fact that any estimate of the positions is affected by an uncertainty (depending on both the dynamic and measurement noises) and therefore any velocity estimate derived from it will always differ from its true value.

Inaccurate velocity estimations may result in a wrong characterization of some of the clusters (i.e. models) that will be later generated.

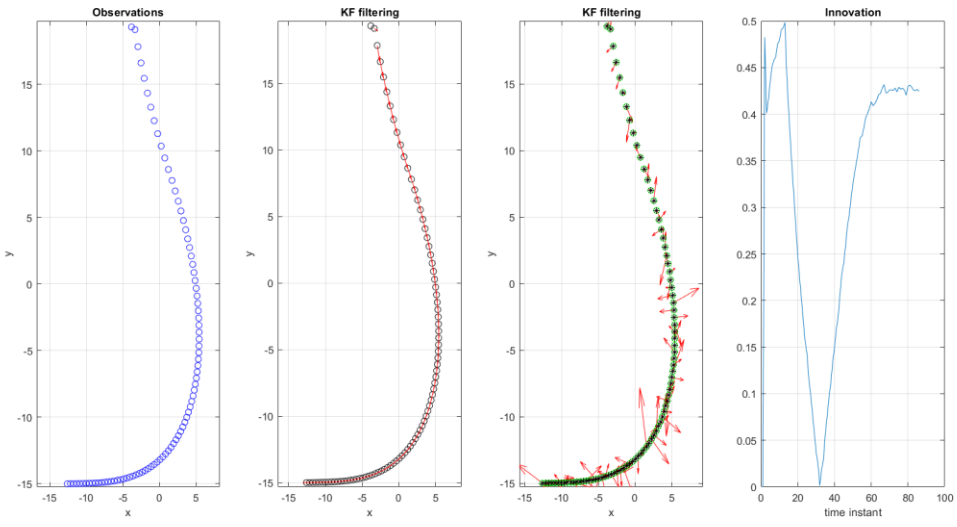


Fig. 4 *Four different graphical outputs of the UMKF for Followers.*

Finally, the fourth plot shows the innovation value throughout all the subsequent filtering steps of the current training trajectory.

However, as the innovation is a 2D element (since it is the difference of 2D position vectors), the plotting is realized only taking into account one of its components (i.e. the one along x).

The innovation shows that the abnormality with respect to the assumption of stillness (null-force field) is reducing as the agent moves towards the attractor point.

From the “mechanical” point of view, the closer the agent comes to the point where the attractor is located the smaller is the potential (i.e. the work still to be done to bring the agent in the position of equilibrium) and therefore the smaller is the speed it can develop.

When the velocity will actually become null (i.e. when the agent will reach the attractor) the null-force assumption will be perfectly appropriate to describe the agent and will cause a reduction of the associated abnormalities.

These assumptions can also be made for the attractor Fig. 1.

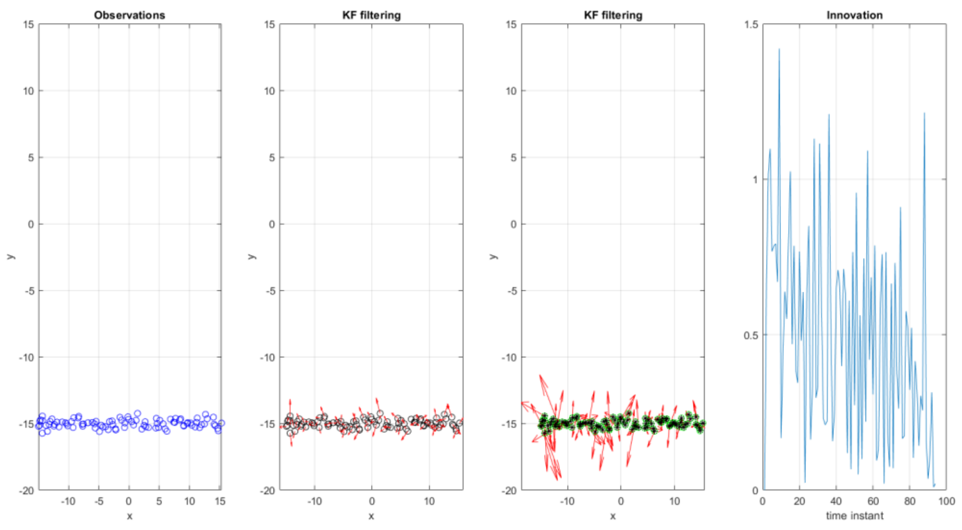


Fig. 5 *Four different graphical outputs of the UMKF for Attractor.*

2.2 Learning $GDBN$ Models

The next step is to learn Generalized Dynamic Bayesian Network ($GDBN$) model [1] (green block in Fig. 3) Step for learning $GDBN$ is show in you need to identify discrete level to make a switching dynamical filtering model.

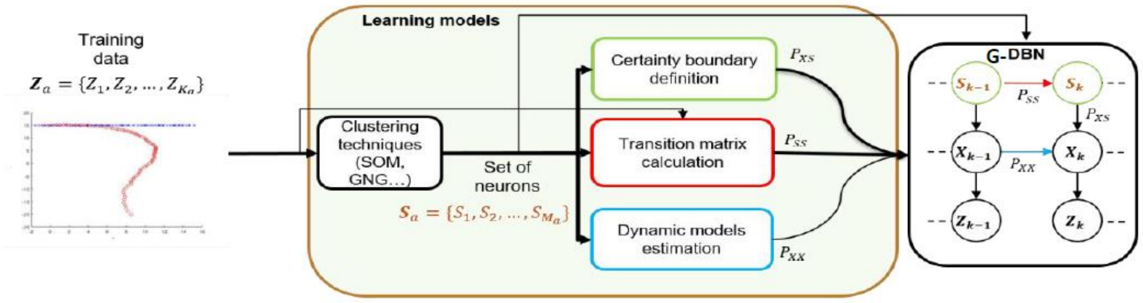

Training data
$Z_a = \{Z_1, Z_2, \dots, Z_{K_a}\}$
Learning models
Clustering techniques (SOM, GNG...)
Set of neurons
$S_a = \{S_1, S_2, \dots, S_{M_a}\}$
Certainty boundary definition
Transition matrix calculation
Dynamic models estimation
P_{XS}
P_{SS}
P_{XX}
G-DBN
S_{k-1}
S_k
X_{k-1}
X_k
Z_{k-1}
Z_k

Fig. 6 *Block diagram for learning models. DBN Dynamic Bayesian Network* [1]

Let $X_K = [x_k \quad y_k \quad \dot{x}_k \quad \dot{y}_k]^T$ be the state of an agent at time k. We model the temporal evolution of the state as quasi-constant velocity behavior locally inside a given region of the environment:

$$X_{K+1} = AX_K + BU_{S_K} + w_K \qquad \text{Eq. 1}$$

Where $S_K \in \boldsymbol{S_a}$; $\boldsymbol{S_a}$ is a set of learned identified regions associated activity a where quasi-constant velocity models are valid, such that:

$$\boldsymbol{S_a} = \{S_1, S_2, \dots, S_{M_a}\} \qquad \text{Eq. 2}$$

M_a is the total number of produced regions for the activity a, see section 2.2.1. Additionally, $A = [A_1 \quad A_2]$ is the transition matrix. We assume no memory to be present in the velocity components: $A_1 = [I_1 \quad 0_{2,2}]^T$ and $A_2 = 0_{4,2}$; $B = [I_2\Delta K \quad I_2]^T$ is a control input model that maps agents' actions (velocities) onto following states, ΔK represents the sampling time; $U_{S_K} = [\dot{x}_{S_K} \quad \dot{y}_{S_K}]^T$ is a control vector containing the expected agent's velocity when its state falls within a discrete region S_K; w_k is the process noise which we assume to be drawn from a zero mean multivariate normal process noise with covariance matrix Q_{S_K}, such that $w_k \sim N(0, Q_{S_K})$. The uncertainty of local state derivatives, Q_{S_K} , depends on the zone where an agent is located.

Let z_k be the observations of an agent's location in a time instant k, such that $z_k = [x_k \quad y_k]^T$. We assume preprocessing step where noisy velocities estimates are computed from initial observations z_k. As a result, a new observation vector Z_k is calculates, such that:

$$Z_K = \left[z_k, \quad \frac{z_{k+1} - z_k}{\Delta k}\right]^T \qquad \text{Eq. 3}$$

This introduce a time delay into processing. As shows in Eq. 1, proposed dynamical models depend on the definition of regions $\boldsymbol{S}_a$, see Eq. 2, which are learned through a discretization process of observed trajectory data $\boldsymbol{Z}_a = \{Z_1, Z_2, \dots, Z_{K_a}\}$, where K_a represents the total number of observations associated to activity a and $Z_k \in \boldsymbol{Z}_a$.
The variable $\boldsymbol{S}_a$, which we call *superstates,* is a discretization of observation $\boldsymbol{Z}_a$. Consequently, superstates $\boldsymbol{S}_a$ are calculate through a weighted distance self-organizing map (SOM).

2.2.1 Self-Organization Map SOM Follower

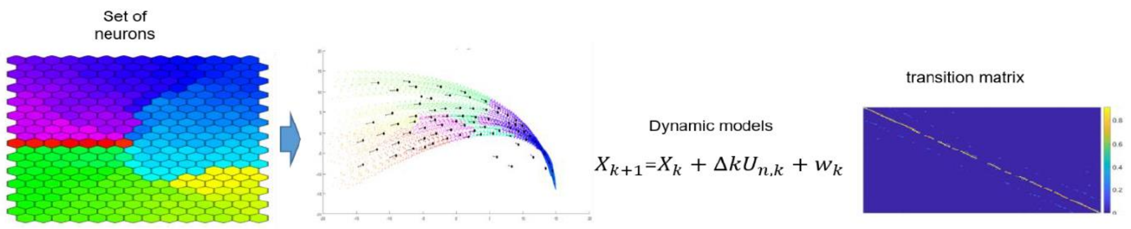


Fig. 7 *Example of learning a switching model*

In Fig. 7 we can see the steps that lead to the dynamic model, from which we extract the transitional matrix.
As unsupervised learning method we use Self-organizing Maps (SOM) [2], [3], [4]. In the proposed SOM procedure, we use two weights, β and α, for the position (x, y) and velocity components $(\dot{x}, \dot{y})$, respectively, where $\beta + \alpha = 1$. We choose $\alpha > \beta$ to favor clustering of patterns with smaller different in speed. Each neuron $S_{m_a} \in \boldsymbol{S}_a$ is a superstate that encodes $\boldsymbol{Z}_a$ into discrete components.
It can be seen that the considered implementation of the SOM algorithm takes into account the "MM" matrix that is the one containing, for each of the samples of each training trajectory, both the estimation of the current position and the innovation (i.e. the error).
MM is a $M \times 4$ matrix with M being total number of training samples. The clusterization will be performed by taking as an input each of its four-dimensional column vectors and classifying it as belonging to one of the N available clusters (i.e. the neurons in the SOM grid).

As already said, the innovation is generated from the difference of the vectors describing the expected (i.e. predicted) and the actual measure for the position of the agent and is assumed as an estimate of its actual (non-null) velocity.
Due to the specific meaning assumed by the error in such vectors, the clusterization will result in the definition of a certain number N of classes whose elements share, for example, some velocity properties and thus allowing the identification of specific dynamic models.
Apart from the previously described matrix of the estimates, the SOM function take in input a sequence of parameters that are used to characterize both the properties and the modalities of the whole clusterization process.
The first two parameters are the scalar quantities α and β that are used to tune the contribution that each of the components of the input vector should have in the definition of the clusters.
In the most general case, that is when α and β share the same value, both the couples identifying the position and the velocity estimates are given the same importance in the characterization of the clusters.
In other words, all the components of the vector affect in a same measure the shift of the firing neuron (i.e. the shift of the set of associated weights) and of the neurons directly connected to it towards the current estimate.
On the contrary, there are cases in which it may be useful to force the influence of one of these information over the other generating, for example, clusters that group together similar kinds of evolutions (i.e. similar velocities) but with no regard on the position from which they generated.
The last group of parameters is made of two scalar values $m1$ and $m2$ and of a third value "SomSize" that is simply defined as their product.
These quantities identify respectively the number of elements of each side of the 2D planar grid of neurons that represent the set of all possible outputs of the clusterization process (i.e. the set of all possible labels that a vector can be given).
The outputs of the SOM function are stored in the "net" structure. The primary output of the clusterization process is contained in the "datanodes" object which is a set of N cells each of them corresponding to a cluster and containing all those vectors that have been associated to it.

A consequent result is the generation of the matrix “Discrete_data” of dimension $Mx1$ whose $i-th$ element is exactly the discrete label of the neuron (i.e. cluster) to which the i-th column vector of the matrix MM has been assigned.
The “Discrete_dataNode” is closely related to the previous output but, instead of listing all the labels in a single structure, it separates them generating a vector of discrete symbols that is specific for each of the training trajectories.
The last object is the “w” matrix that contains the weights of each neuron that are the quantities employed to evaluate which of the labels is the most suited to be chosen (i.e. which of the neurons has the highest response).
This matrix represents only the final set of weights obtained through subsequent adaptations (generated by each new classified vector).
It has already been said which is the role of the two parameters α and β in the separation of the N clusters and in the definition of their location and shape.
At the beginning, α and β are both set equal to 0.5 to let the SOM algorithm work under the assumption that there should be an equilibrium in their contributions during clusterization.
The result will be the generation of clusters specific for a certain zone of the xy plane but, at the same time, a same zone will be “vertically” subdivided in groups that consider the different magnitude values of its associated velocity vectors.
Obviously, it is possible to specify many different clusterization logics by simply tuning the two scalar parameters in an appropriate way.
When setting, for example, α equal to zero and keeping β unchanged, the grouping into different classes will consider the only the position information that is carried in the first two components of the vector.
Consequently, it is reasonable to expect that the SOM algorithm will generate clusters that aggregate vectors with similar locations but having velocities that are potentially distributed along their whole domain (i.e. no discrimination between clusters is realized in this “vertical” direction).
In fact, from the Fig. 8 -Fig. 9, it is possible to identify a set of clusters (i.e. colours) each one covering a wide set of velocity values.

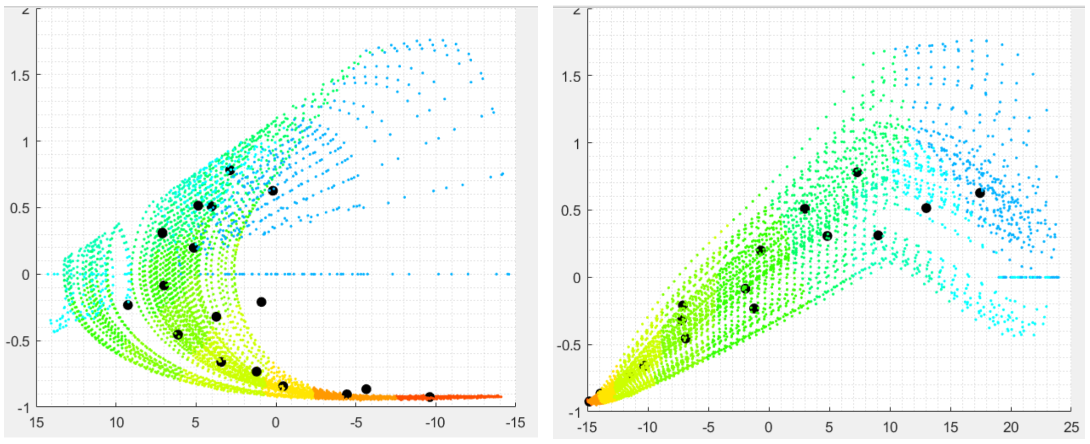


Fig. 8 *Representation of the input vectors and of the weights of each neuron projected onto the $x\dot{x}$ plane (on the left), onto the $y\dot{x}$ plane (on the right)*

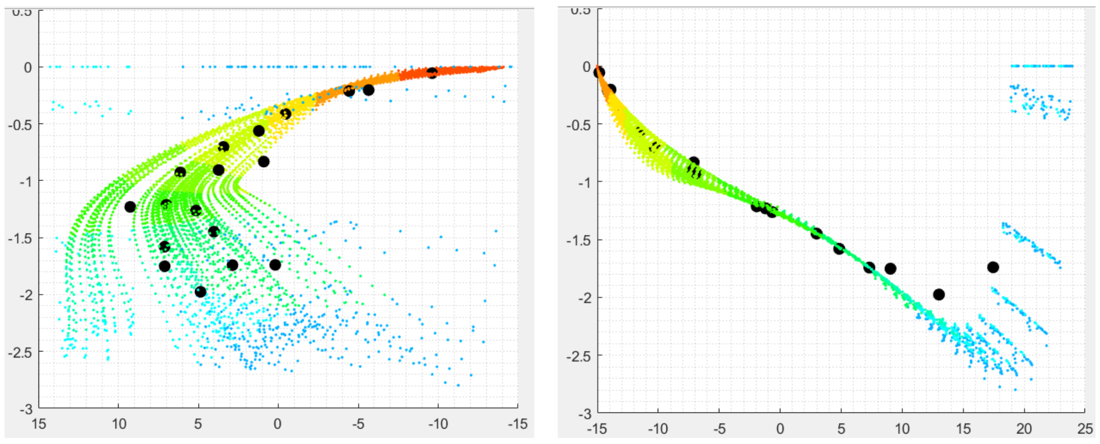


Fig. 9 *Representation of the input vectors and of the weights of each neuron projected onto the $x\dot{y}$ plane (on the left), onto the $y\dot{y}$ plane (on the right)*

The cases in which different clusters (i.e. different colours) are displayed in a same column have nothing to do with the differences in the velocity values.

This situation must be referred to the fact that, although the vectors share the same component along x, they can still be featured with very different y values and therefore they can still have very different positions in the xy plane.

On the contrary, when setting α equal to one and decreasing β, the clusterization tends to neglect the displacement of the vectors in their x and y components and generates classes considering more the magnitude of their associated velocity.

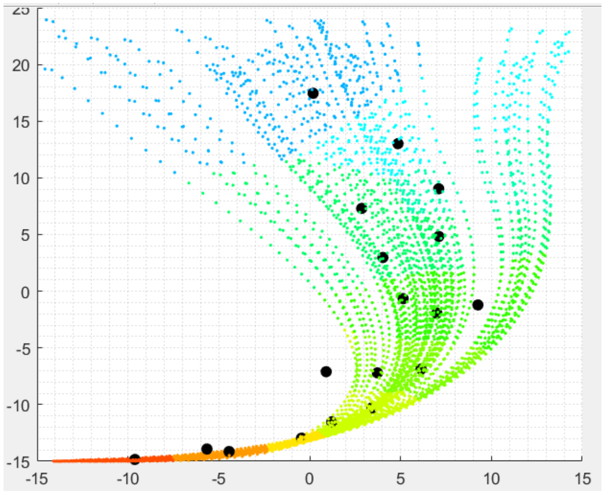


Fig. 10 *Representation of the input vector and of the weights of each neuron project onto the xy plane.*

As a consequence, a very wide range of positions is gathered in a same group and, at the same time, many clusters are overlapping in a same area of the xy plane indicating that they are separating groups along the z component only (i.e. one of the components of the velocity).

Modifying the total number of neurons changes the number of clusters to which the vectors can be associated.

Therefore, there will be a variation in the distance between the distributions of vectors belonging to adjacent classes.

In other words, the higher the number of neurons the more accurate (i.e. detailed) can be the classification of the data as each label will be associated to a smaller and more specific group.

Different figures are displayed by the SOM (script) implementation visualizing information about both the scheme used for the clusterization and the obtained results.

The first and second figures (Fig. 11 - Fig. 12) both represent each of the N available neurons through hexagons. In the first figure each of the hexagons is associated to a quantity that represents the number of different vectors that have been assigned to that specific neuron.

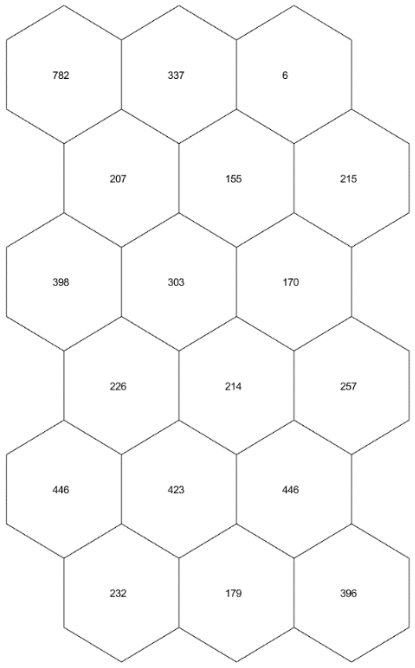


Fig. 11 *Neurons (i.e. the hexagons) containing the number of input vector assigned to each of them.*

In other words, each hexagon displays how many times the input vector has caused that neuron to respond with a greater intensity than all the others.
In the second figure, this information is just transferred, through a specific map, into a colour that is used to fill the hexagon.

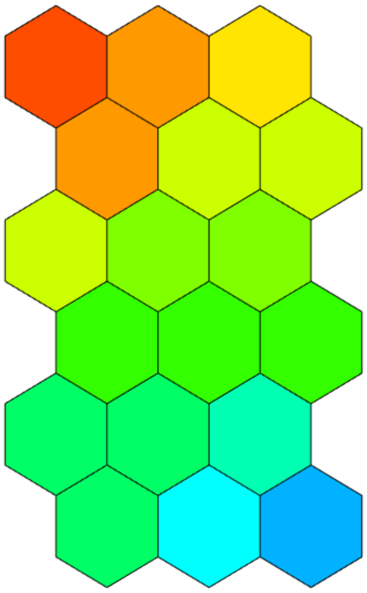
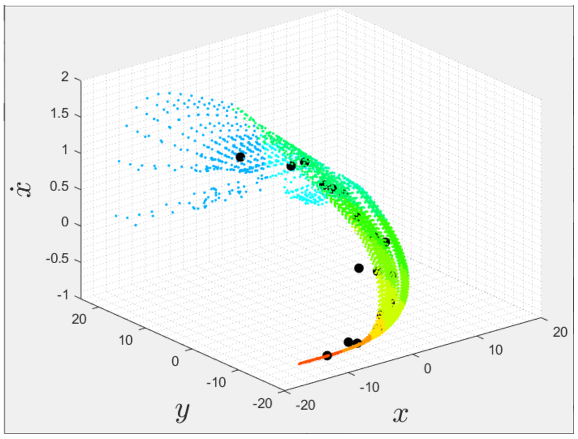


Fig. 12 *Neurons (i.e. the hexagons) filled with the colour identifying the number of their element*

The last two figures (Fig. 13) are used to visualize the three-dimensional scatters that locate each input vector in the 3D space through its position (along the x and y axes) and one of the two components of its velocity (along the z axis).
Moreover, each of the scattered points is given a certain colour that identifies the specific neuron to which it has been associated.
Together with the input vectors, also the weights (or at least three of their four components, because you cannot visualize 4 dimensions) of each neuron are scattered in this same 3D space to allow the visualization of the final (i.e. after the end of the SOM algorithm) distribution of the neurons (i.e. the one that allows the most efficient classification).



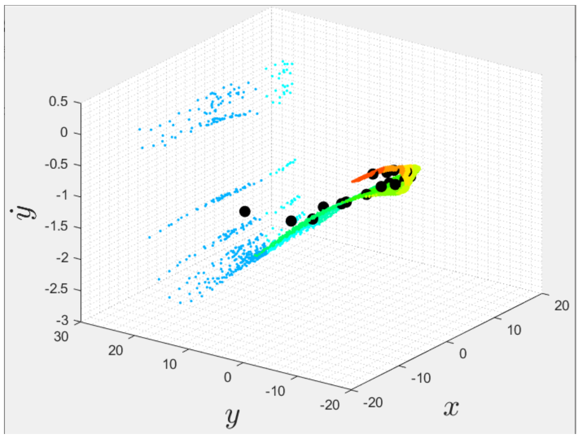


Fig. 13 *Representing of the input vectors in three-dimensional space taking into account only the $\dot{x}$ (on the left) and $\dot{y}$ (on the right) component of the velocity vector respectively.*

In the case of followers, the parameters for the SOM are the following:

Parameter	value
α	0.80
β	0.20
m_1	6
m_2	3

2.2.2 Dynamic Models (vocabulary)

From the clustering of dataset, you are now at discrete level. Now the next step is to compute the dynamic model Eq. 1 which define the properties of cluster such as mean, covariance, boundary of cluster etc.

The generation of the vocabulary coincides with the extraction of the characteristic properties (P_{XX}, P_{XS}, P_{XX}) [1] of the clusters that are required to convert the results of the clusterization into the definition of an actual $MJPF$ and the characterization of its dynamics (both continuous and discrete).

On this purpose, the first functions compute the mean and covariance of the vectors belonging to a given cluster with the objective to provide an average associated behavior and its variability (Certainty Boundary Definition P_{XS}).

On the contrary, the last two functions are used to derive the discrete dynamics that characterize the succession of the different motion modalities in time (Transition matrix calculation P_{SS} and Dynamic model estimation P_{XX}).

Starting from the set of labelled data derived from the application of the SOM clustering to the input vectors, the function “GetTransitionMatrix” can simply count, for each cluster, how many times its label n persists through two consecutive iterations or how many times a certain change (from n to m) occurs.

By normalizing the accumulated values, a row of the transition matrix is obtained.

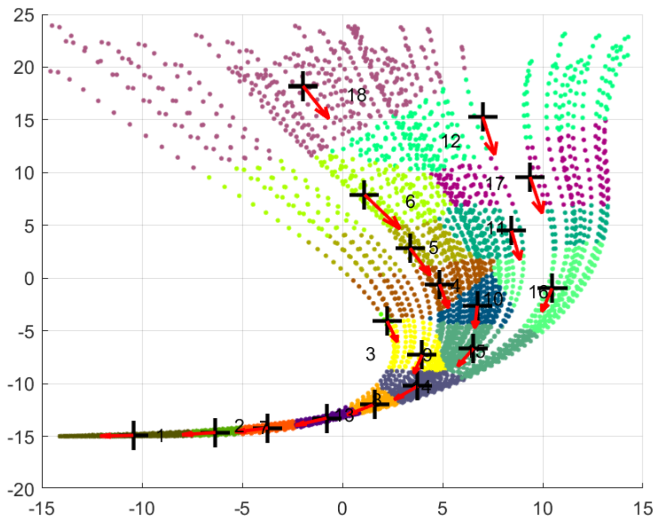


Fig. 14 *clasturized trajectories*

In Fig. 14 we can see the mean position of cluster $(identify\ by\ the\ cross\ +)$ and their mean velocity $(identify\ by\ the\ red\ arrow\ \rightarrow)$. Every point with the same colour belongs to the same cluster.

2.2.3 Transition Matrix Follower

The following images show the transition matrix for followers Fig. 15 and attractors Fig. 19. In the systems considered until now, the quantity to be estimated was represented by the state x_k of a certain agent that evolved in a continuous way in time.

Now, the evolution of the system is described from a higher level through the use of a discrete variable x_k describing the position of the agent in a more generic way without having to specify the couple of coordinates (x, y).

In this particular case, the label that is given to the agent at each time step is used to identify whether it belongs to one or another of the areas in which the space has been divided through a process of clusterization.

For this reason, the probability $p(x_k/x_{k-1})$, which represents the information borrowed by the previous state onto the next one, can only be computed for the set of combinations of the of the symbols that x_k and x_{k-1} are allowed to assume.

In other words, if m is the cardinality of the set of symbols that can be associated to each of the discrete variables, a square matrix Π can be defined containing the collection of all the mxm probabilities $p(x_k = \alpha_i/x_{k-1} = \alpha_j)$ with α_i and α_j belonging to the same set of symbols $\{\alpha_1, \alpha_2, \dots, \alpha_{m-1}, \alpha_m\}$.

$$\Pi = \begin{bmatrix} p(\alpha_1/\alpha_1) & p(\alpha_2/\alpha_1) & \dots & p(\alpha_{m-1}/\alpha_1) & p(\alpha_m/\alpha_1) \\ p(\alpha_1/\alpha_2) & p(\alpha_2/\alpha_2) & \dots & p(\alpha_{m-1}/\alpha_2) & p(\alpha_m/\alpha_2) \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ p(\alpha_1/\alpha_{m-1}) & p(\alpha_2/\alpha_{m-1}) & \dots & p(\alpha_{m-1}/\alpha_{m-1}) & p(\alpha_m/\alpha_{m-1}) \\ p(\alpha_1/\alpha_m) & p(\alpha_2/\alpha_m) & \dots & p(\alpha_{m-1}/\alpha_m) & p(\alpha_m/\alpha_m) \end{bmatrix}$$

This Π matrix is called “transition matrix” as it expresses the discrete dynamics of the agent defining the probabilities associated to each possible transition from one label to another in two consecutive time steps $k-1$ and k.

In the proposed examples, the transition matrix is presented to describe the evolution of the label variable in a discrete domain made of $m = 18$ distinct symbols respectively (18 different cluster):

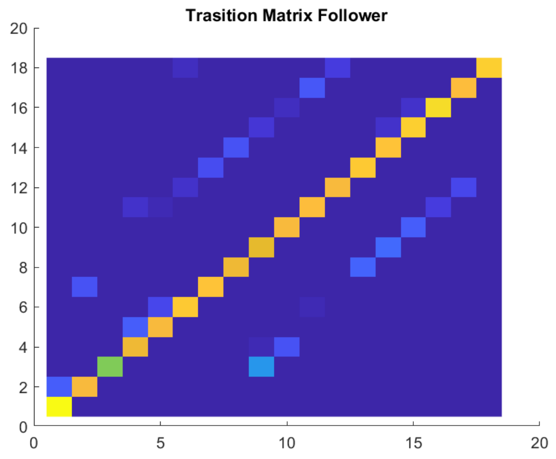


Fig. 15 *Transition Matrix Follower*.

The use of the function “imagesc” is convenient for obtaining a graphical representation Fig. 15 of these kind of matrixes as it allows the generation of an image where the information contained in the transition matrix is translated into the color properties of the figure itself.
The result will be an image where the (i, j) element is a uniform squared area whose colour is the interpretation (through a certain colour-map) of the magnitude of the corresponding (i, j) component of the matrix.
In other words, the event of having a transition from the label α_i to α_j has a very high probability in all those cases where $i = j$ meaning that the most common case, whatever is the current label, is to have no transitions at all (i.e. staying in the current cluster in two consecutive instants $k - 1$ and k).
The transition matrix can be considered as a compact representation of an HMM model.
Each of the m nodes in the graphical HMM representation is associated to a specific row of the matrix Π and each of the m links connecting it to other nodes is associated to a column (that contains the corresponding value of the transition probability).
Due to the discrete nature of this variable representing the highest hierarchy in a two-layer DBN, the HMM model represents the most appropriate tool to be employed to realize inferences on the future condition of the system.

In fact, as already said, such a model, and its associated transition matrix, contains all the information concerning the probabilistic properties of the dynamics of the agent as it evolves from stage $k-1$ to stage k.

2.2.4 Self-Organization Map SOM Attractor

To describe a method to learn a new $GDBN$ some hypothesis must be done on attractor composing the mixture. In particular, the discussion will be focused on the case where a class of potential function $\varphi^{\boldsymbol{i}}(x) = (x - m_i)^2$ will be chosen where m_i is chosen as a geometrically defined subspace of X.

Let m_i i.e. the attractor of the $i - th$ component of the Mixture be formed by a set of attractor points on a linear subspace of state space X.

A linear subspace $\Psi_i \in X$ can be defined as the set of points y such that $\mu_i y + b_i = 0, y \in X$ i.e. a straight line. This linear subspace contains a subset of attractor points, $y = m_i$, characterizing component i_{th} of the mixture.

A single point $y_i(x) = m_i$ in linear subspace I should be associated with a point x of the state space to define a potential field covering in a dense way the state space X.

Function $y_i(x)$ should be defined in such a way to be valid for all linear subspace could be defined. $y_i(x)$ should be such that $\mu_i y_i(x) + b_i = 0$ as the linear subspace is the locus of attractors for component i of the mixture and the potential function can so be written as:

$$\varphi^{\boldsymbol{i}}(x) = \left(x - y_i(x)\right)^2 : y, x \in X \qquad \text{Eq. 4}$$

Attractors are associated with a velocity flow:

$$f(v) = (y_i(x) - x) \qquad \text{Eq. 5}$$

The flow $f(v)$ describes velocities fields and the function $y_i(x)$ has to define a property describing how the law of motion applies on the linear subspace. Each couple of points, $\left(x, y_i(x)\right)$ should be such that the velocity component v at x should be directed towards a point $y_i(x)$ on the linear subspace.

In this case (attractors) the SOM parameters are:

Parameter	value
α	0.80
β	0.20
m_1	2
m_2	4

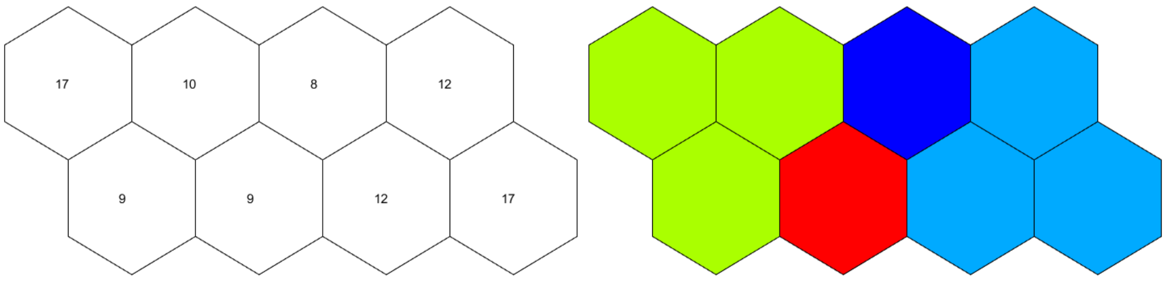


Fig. 16 *Neurons (i.e. the hexagons) containing the number of input vector assigned to each of them (on the left), filled with the colour identifying the number of their element (on the right)*

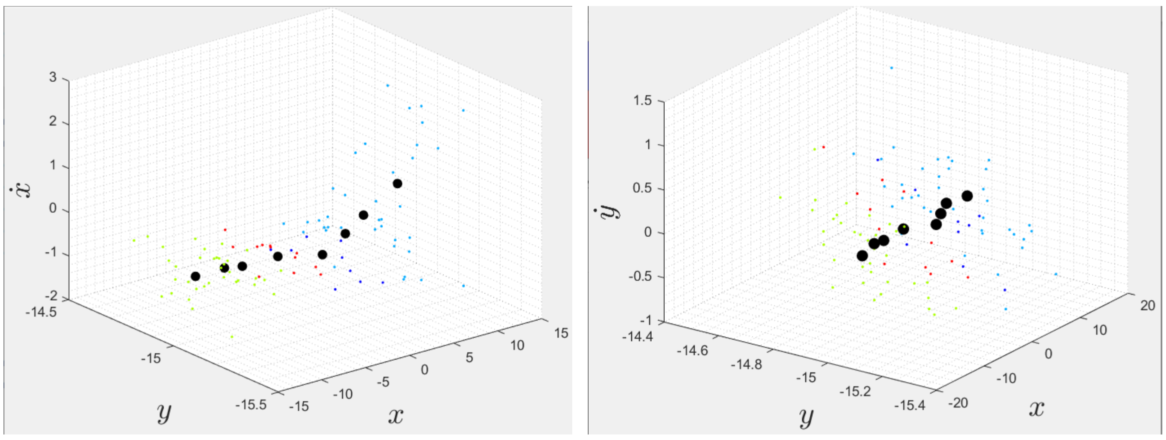


Fig. 17 *Representing of the input vectors in three-dimensional space taking into account only the $\dot{x}$ (on the left) and $\dot{y}$ (on the right) component of the velocity vector respectively.*

As said previously, each of the scattered Fig. 17 points is given a certain colour that identifies the specific neuron to which it has been associated.

Now the next step is to compute the dynamic model Eq. 1 which define the properties of cluster such as mean, covariance, boundary of cluster etc.

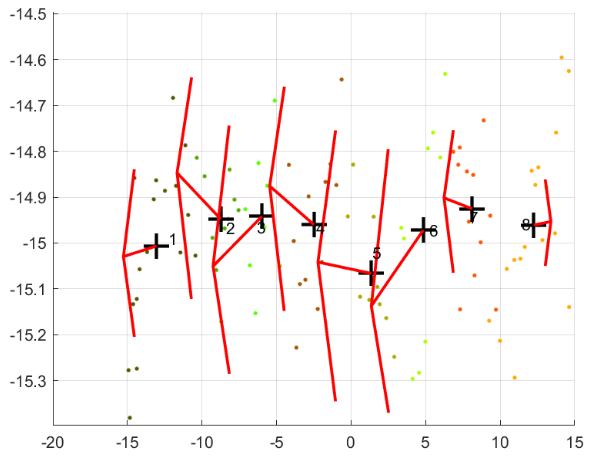


Fig. 18 *clasturized Attractor*

In Fig. 18 we can see the mean position of cluster ($identify\ by\ the\ cross +$) and their mean velocity ($identify\ by\ the\ red\ arrow \rightarrow$).

Also, in this case we have a representation of the transition matrix

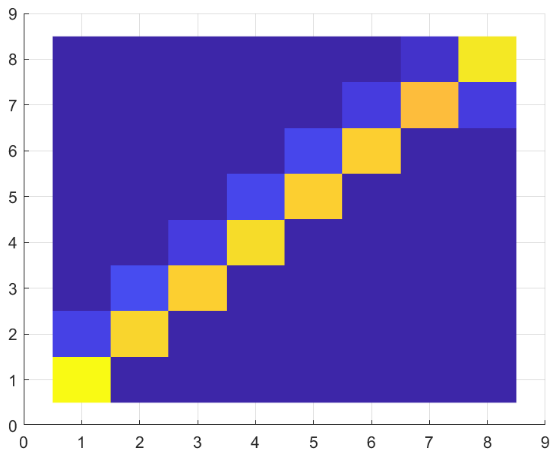


Fig. 19 *Transition Matrix Attractor.*

This set of information will be important in the following, in case you want to study the abnormalities during the iteration between the followers and the attractor.

3. Testing phase

The dataset provided for the testing phase is defined in the section 1.2. In the following section we generate generalized states for testing dataset using $UMKF$, similar to the training phase. While in section 3.2 there will be a discussion about the $MJPF$.

3.1 Null Force Filter

From the clustering (section 2.2) we get a dataset, the available dataset is made of a group of measures derived, through a specific measurement model, from different examples of trajectories that the agent can follow when subject to a certain set of rules (i.e. dynamic models).

As it can be seen from the scattering of such data, the rules embedded in these trajectories Fig. 2 differ from those that were obtained from the clusterization of the generalized state produced when the system was characterized by the attractor only Fig. 1.

In fact, in the ordinary case, the acceleration (and therefore the movement) provided to the agent could only be directed towards the spot occupied by the attractor (i.e. along the steepest direction of the potential) as this was the only component impressing a force.

In the considered case, some other element seems to have an influence on the agent since its movement Fig. 2 cannot be explained anymore by the single force field generated by the attractor.

During its estimation process, the $MJPF$, trained with the old data, is used to generate new innovation values that are employed to identify abnormalities and explain when the current trajectories cannot be explained in terms of the old models and therefore when the introduction of new dynamics is required.

The generalized state is defined through the computation of the error (projected into the measurement space) that is associated to the null-force prediction and contained in the innovation vector.

In a normal scenario, only one component is influencing the agent and its dynamic evolution has to be referred only to the potential energy provided by the attractor.

Since the bell-shaped potential energy has its center in the position m occupied by the attractor, the agent, independently from its current location x, will gain a velocity (i.e. it is given a certain acceleration) in the direction described by the vector $m - x$.

In other words, the agent starts moving along the descent direction of the gradient towards the potential minimum (i.e. towards the attractor).

When an obstacle capable of imposing a certain repulsive force is introduced in the system, a new peak will be added to the potential (due to the superposition of the effects) that will cause a deviation in the trajectory that the agent follows while descending the gradient (depending on the initial position of the agent, a normal path towards the attractor may not be possible anymore).

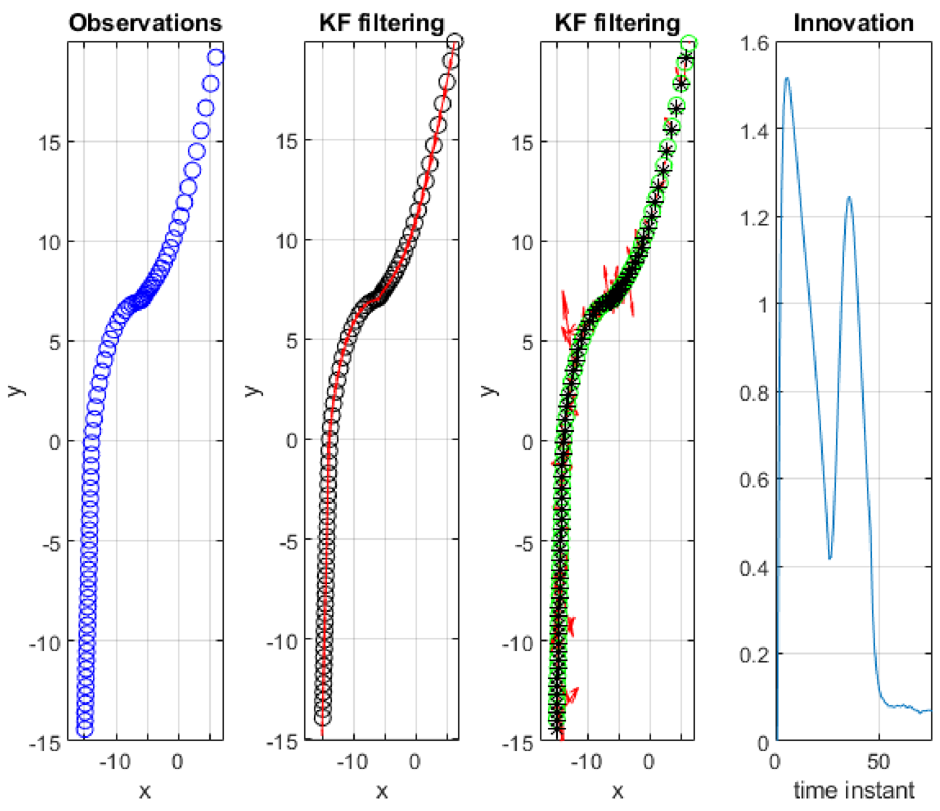


Fig. 20 *Four different graphical outputs of the UMKF follower test*

It has already been said that the use of the UMKF, through the particular choice on the supposed dynamic model (i.e. static), allows the generation of innovation vectors that coincide with the difference between the last estimated position and the observed current position.

Therefore, the filter is able not only to generate the estimation of the actual position (through the update and employing the evidence carried by the observation) but also to associate to this estimation the non-null velocity that has caused the shift.

The result is the generation, for each of the tasting trajectories and for each of their elements, of the four-dimensional vector of the "generalized states" containing both the position of the agent and its first order derivative.

3.2 Markov Jump Particle Filter ($MJPF$)

In order to make inference on the learned $GDBN$, we can use the $MJPF$ (white block in Fig. 3) Abnormalities measurements will be its output (orange block in Fig. 3).

In Jump Marko Process (JMP) estimation of posterior is obtained by using combinations of inference approaches introduced for two level $DBNs$, i.e. KF, HMM and $Particle\ filters$.

$MJPF$ is a mix between Particle filter and Kalman filter, this works for linear and Gaussian case.

$GDBN$ for $MJPF$ is shown below Fig. 21. This represents a hybrid model: continuous and discrete hidden variables

- Continuous variables have conditional linear Gaussian dynamic
- Discrete variable has no continuous parameters

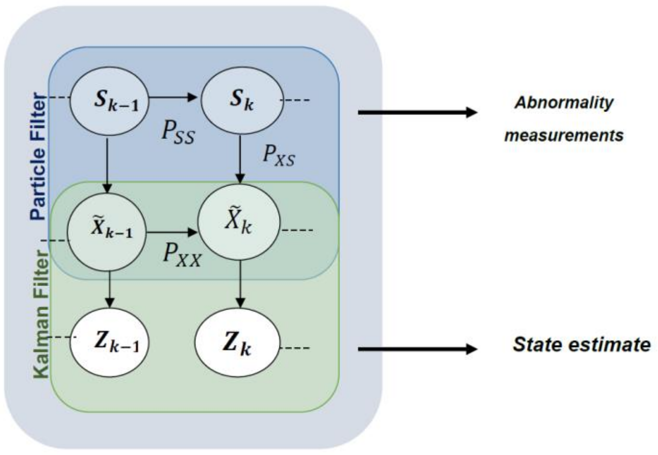
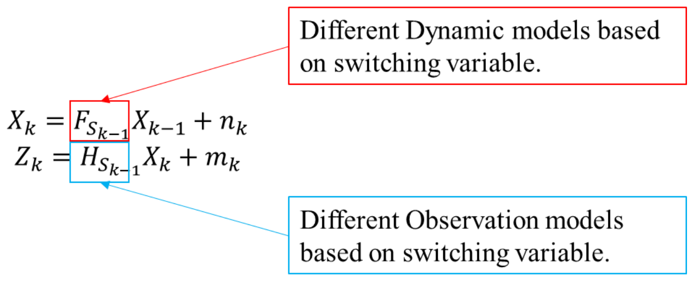


Fig. 21 *Generalized Hierarchical Dynamic Bayesian Network* ($GDBN$) *for* $MJPF$

Where Z_k is the measurement, $\tilde{X}_k$is the state and S_k is the superstate.

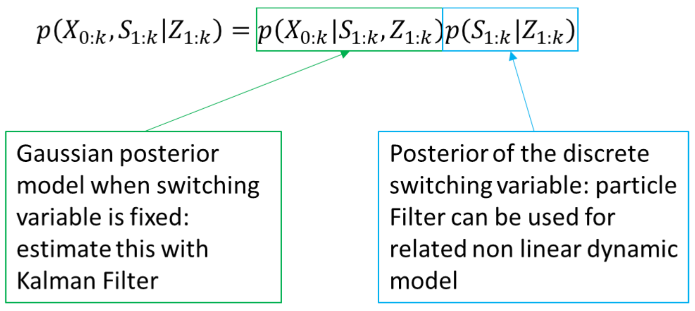
In our case, $\tilde{X}_k$ is made up of position and velocity of the object and constitute the continues level, approximated by a Kalman Filter. The superstate is a discrete variable that is approximated by a Particle Filter



Different Dynamic models based on switching variable.
$X_k = F_{S_{k-1}} X_{k-1} + n_k$
$Z_k = H_{S_{k-1}} X_k + m_k$
Different Observation models based on switching variable.

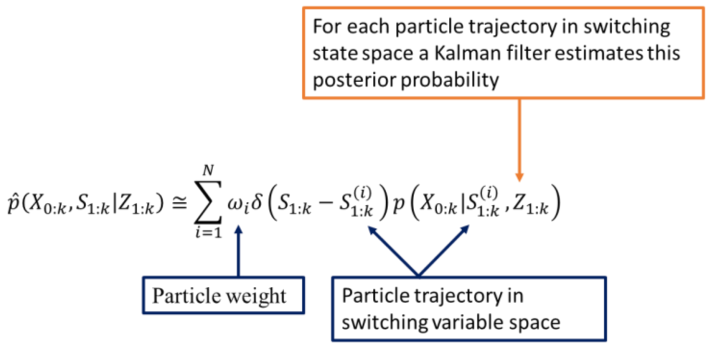
Fig. 22 *Contextually linear models with Gaussian noise*

$MJPF$ idea:



$p(X_{0:k}, S_{1:k}|Z_{1:k}) = p(X_{0:k}|S_{1:k}, Z_{1:k}) p(S_{1:k}|Z_{1:k})$
Gaussian posterior model when switching variable is fixed: estimate this with Kalman Filter
Posterior of the discrete switching variable: particle Filter can be used for related non linear dynamic model

$PDFs$ can be represented in the following way [5]:



For each particle trajectory in switching state space a Kalman filter estimates this posterior probability
$\hat{p}(X_{0:k}, S_{1:k}|Z_{1:k}) \cong \sum_{i=1}^{N} \omega_i \delta\left(S_{1:k} - S_{1:k}^{(i)}\right) p\left(X_{0:k}|S_{1:k}^{(i)}, Z_{1:k}\right)$
Particle weight
Particle trajectory in switching variable space

Where $p\left(X_k \middle| S_{1:k}^{(i)}, Z_{1:k}\right) = N\left(\hat{X}_{k|k}^{(i)}, P_{k|k}^{(i)}\right)$ is defined as a normal distribution with $\hat{X}_{k|k}^{(i)}$represent mean and $P_{k|k}^{(i)}$ variance. Prediction part of Kalman filter depends on switching variable as:

$$\hat{X}_{k|k-1}^{(i)} = F_{S_{k-1}} \hat{X}_{k-1|k-1}^{(i)}$$
$$P_{k|k-1}^{(i)} = F_{S_{k-1}} P_{k-1|k-1}^{(i)} F'_{S_{k-1}} + Q$$

For abnormality measurements, probabilistic distances such as Bhattacharyya distance as show in Eq. 6 is used at both continuous and discrete level:

$$D_b(N_1, N_2) = \frac{1}{8}(\mu_1 - \mu_2)^T \Sigma^{-1} (\mu_1 - \mu_2) + \frac{1}{2} \ln \left(\frac{\det \Sigma}{\sqrt{\det \Sigma_1 \det \Sigma_2}} \right) \quad \text{Eq. 6}$$

$$where\ \Sigma = \frac{\Sigma_1 + \Sigma_2}{2}$$

You will be able to measure abnormalities called errors by using $MJPF$. So, from these errors which will be an incremental learning to enhance the self-awareness of agent as shown in Fig. 3.

In this case, three types of measurement are used:

- Innovation, this value was computed as a simple Euclidean distance between two multidimensional vectors (i.e. prediction and observation).
- $db1$ measure the Bhattacharyya distance between the distribution of the prediction and the likelihood of the discrete state.
- $db2$ measures the Bhattacharyya distance between the distribution of the prediction and the likelihood of the continuous state.

3.2.1 Training phase

Initially, the MJPF is trained with the vocabulary provided by the clustering phase in the section 2.2.1

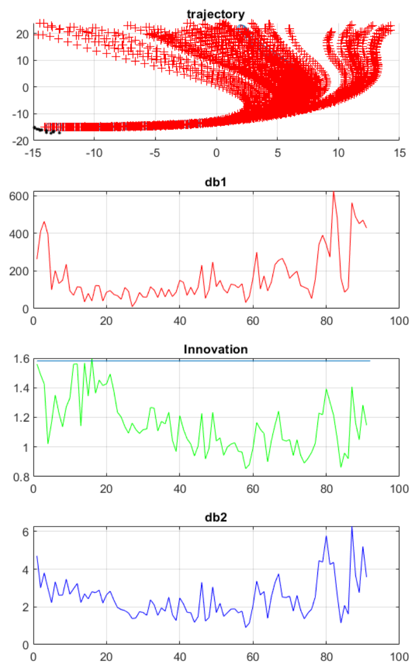


Fig. 23 *measures used in the MJPF (Training phase)*

During the training phase, a threshold value (Fig. 23 in innovation) is obtained which allows us to evaluate abnormalities in testing phase.

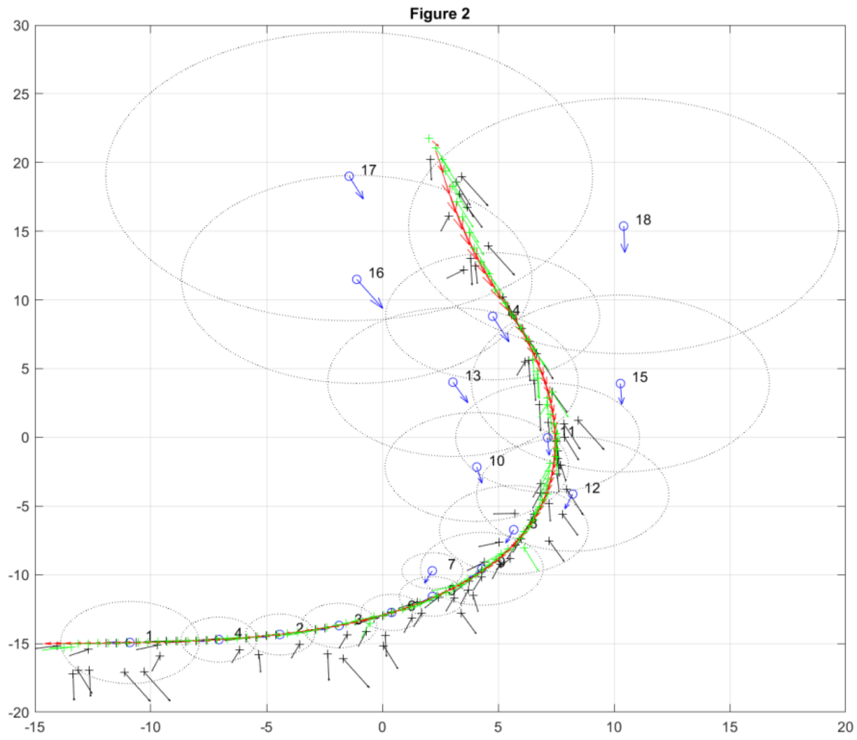


Fig. 24 *Clusters and train trajectory*

3.2.2 Testing phase

When the training phase is over, the test trajectories are input to the $MJPF$ (which have been generalized with the UMKF in section 3.1), In this way the abnormality can be calculated.

In Fig. 25 we have the following information:

The blue stars: testing trajectories.

Green arrows: velocity of the updated state

Green points: position of the updated state

Black arrows: velocity of the predicted state

Black points: position of the predicted state.

Red arrows: velocity vectors of testing data

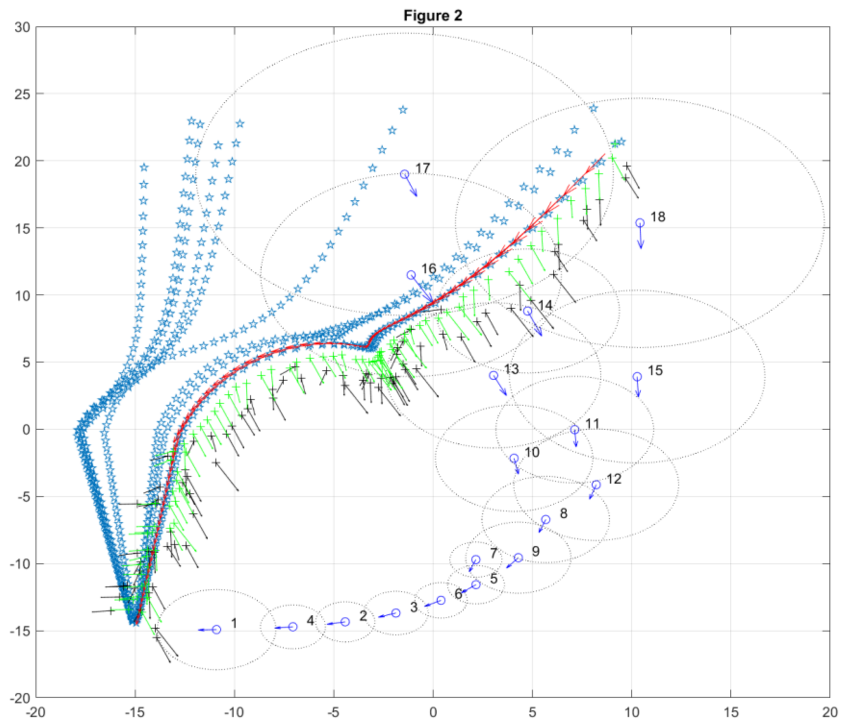


Fig. 25 *Clusters and test trajectories*

In this particular case, however, the $MJPF$ that performs prediction has not been trained on the current generalized states (embedding the rules associated to the obstacle) but simply uses them as input vectors on which it will be able to evaluate its prediction accuracy for both position and velocity.

Therefore, it is possible to expect that the filter will still generate high abnormality values whenever the input generalized vector describes a behavior that has a non-negligible contribution coming from the obstacle.

This is what happens in the abnormal scenario where the testing set for the $MJPF$ is made of the generalized states that were obtained by applying the $UMKF$ to one of the trajectories affected by the influence of the obstacle.

Until the agent remains far from the obstacle, the repulsive contribution can be neglected, and the movement can be thought as characterized by the action of the attractor only.

This imposes a force and causes a modification of the position of the agent according to a set of dynamics that are well known to the $MJPF$ (i.e. previously learned and currently adopted by the filter).

When the agent gets closer to the obstacle and its influence becomes stronger, the testing trajectory deviates.

Whatever is the assumption on the next cluster (i.e. multinomial extraction through the use of the transition matrix) for the current particle, the associated dynamic model will be insufficient for a correct description of the agent causing an increase in the difference(i.e. innovation) between the average (among all particles) predicted position and the measured one.

The same phenomenon, signaling the presence of abnormalities, can be identified in the $db2$ plot that represents not just an Euclidean distance between two position vectors but a measure of the difference existing between two (gaussian in the case of $MJPF$) distributions associated to the prediction and to the generalized state (i.e. position and velocity) respectively.

The abnormality has obviously to be referred to the fact that the whole vocabulary employed by the $MJPF$ was generated in a different context where the velocity associated to each cluster just expressed the average speed imposed by the force field of the attractor in that specific area of the space (i.e. space dependent dynamic model).

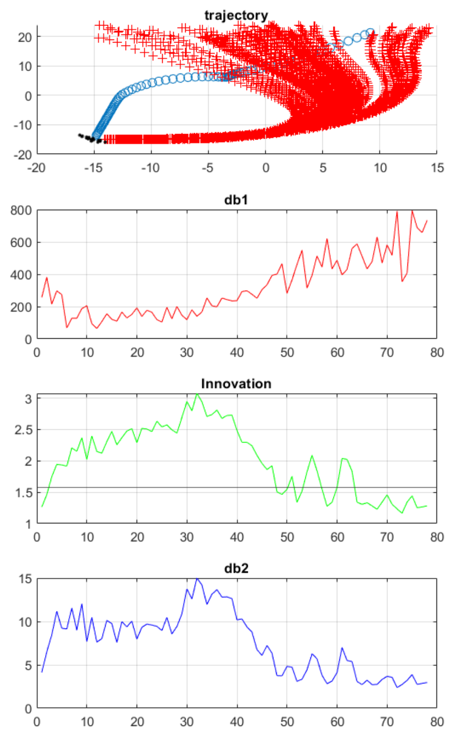


Fig. 26 *measures used in the MJPF (Testing phase)*

In the first plot in Fig. 26, it is possible to identify three different kind of information.
The first one, displayed through the points scattered in red, represents the training trajectory.
This is the trajectory that, through the clusterization process (i.e. aggregation of the observations in groups), allows the generation of a set of m clusters with a certain position and a certain velocity (dynamic) model both obtained by averaging the values of their elements.
In other words, it is the set of data from which to extract those evolution modalities that will be later employed in the $MJPF$ switching model and that will represent the only source of knowledge available to the filter in performing its prediction.
The second group of points, the blue one, represents the set of observations of a certain trajectory followed by the agent that is employed as a test trajectory to evaluate the filter tracking performances.
In other words, it coincides with the sequence of measurements z_k that will be used to correct (update) the estimation performed by the MJPF when it is trying to predict the agent evolution with one of the available dynamic models.
The last set of information, that is given by the group of black points, is not distributed along the whole trajectory but its concentrated around the position of the agent observed at a given stage k.
In fact, these points represent the prediction that each particle performs on the state of the agent at the next step obtained by applying one of the feasible models as described by the discrete dynamics contained in the transition matrix.
In other words, at the stage $k - 1$ a set of N particles is given each one associated with a label (i.e. the cluster) identifying the specific dynamic model that the agent has followed to reach its actual state.
For each of them, a new symbol is extracted from a multinomial distribution (built by employing the probabilities of transition taken from the associated row of the matrix) and it represents one of the possible changes in the modality with which the agent moves.
By applying each of these models, a particular prediction on the state of the agent at stage k is generated that will correspond to one of the points scattered in the plot.
The fact that they are almost all concentrated instead of being sparse is due to the fact that after resampling many of the particles will coincide.

Therefore, all of them will start from a same current state and will use the same dynamic model to perform the prediction thus generating inferences that will differ only due to the noise component affecting the model itself.

In this particular case, the testing trajectory on which to perform the estimation is characterized by some behaviors that appear to be extraordinary when compared to the set of feasible and expected ones.

A particle that is extracted from the multinomial distribution (and representing the predicted modality for generating the next inference), can only be associated to one of those models (i.e. labels) that the filter has learned to represent a feasible transition from the current cluster (as described by the corresponding row of the transition matrix).

In other words, if the testing trajectory is featured with a change in the evolution modality that was not previously identified (i.e. learned) as typical (i.e. possible) for the current cluster, none of the particles will employ an appropriate model in the generation of the prediction.

For the whole period in which the agent continues to move according to the unknown dynamics, even the particles with the highest weight (those identified as the best in performing the prediction) will generate an inconsistent inference with respect to the observation z_k.

This will cause a progressive increase of the distance between the Gaussian pdfs of the predicted state and of the state evidence coming from the observation.

Therefore, whenever a behavior that has never been observed before or alternatively that cannot represent a possible successor of the current one (according to what is stated by the transition matrix) appears in the test trajectory, the $MJPF$ will give a set of predictions that are all generated from uncompliant models.

As a result, the abnormality parameters will assume high values for all the particles since none of them has the possibility to infer with an appropriate prediction model.

It is known that the $MJPF$ evaluates the coherence existing between the prediction realized through the dynamic model hypothesized by each particle and the observation.

This coherence can be interpreted as the evidence carried by the measure in favor of a given model and therefore it can be employed in the computation of the weights.

The considered $MJPF$ realizes the weighting and therefore the validation of the models proposed by each particle by taking into consideration two new estimators instead of the only innovation parameter.

Both these quantities are defined as Bhattacharyya ($db1\ and\ db2$ in Fig. 26) distances meaning that their magnitude is an interpretation (through a specific formula Eq. 6) of the difference between two distributions.

The $db2$ distance is very similar to the innovation parameter as its objective is to measure the inconsistency of the inference with respect to the observation.

However, the innovation parameter only dealt with the mean value of the Kalman inference, this value was computed as a simple Euclidean distance between two multidimensional vectors (i.e. prediction and observation).

On the contrary, the Bhattacharyya distance is dealing with whole distributions thus taking into account not just a single predicted value but also the uncertainty that is featured with that specific choice.

In other words, if both the prediction and observation are distributed as gaussians (true due to the linearity of the dynamic and measurement models), the final coherence evaluation will be a combination of the information regarding the distance of their centers (i.e. mean points) and their covariances.

It has already been said that each cluster is an aggregation of the samples of a training trajectory grouped according to some of their features.

Therefore, by considering all the elements of such groups, it is possible to compute an estimate of the mean and the covariance characterizing each cluster.

The $db1$ distance uses these parameters to describe the difference between the prediction distribution and the samples distribution (assumed gaussian) for the cluster from which the current inference model was extracted.

The further the prediction is moving from the center of a cluster the less the model that generated it can be trusted and the smaller should be the weight associated to the corresponding particle.

In order to distinguish in an automatic way abnormal scenario, we need to identify a threshold (obtained in the training phase) for the innovation see Fig. 26Fig. 26. If the innovation is under the threshold, measurement represent a normal case and so the model approximates well data, if the innovation is over the threshold, measurements represent an abnormal case and so the model is not a good approximation for data.

These abnormalities can be saved to generate new sequences based on these generalized errors see Fig. 27:

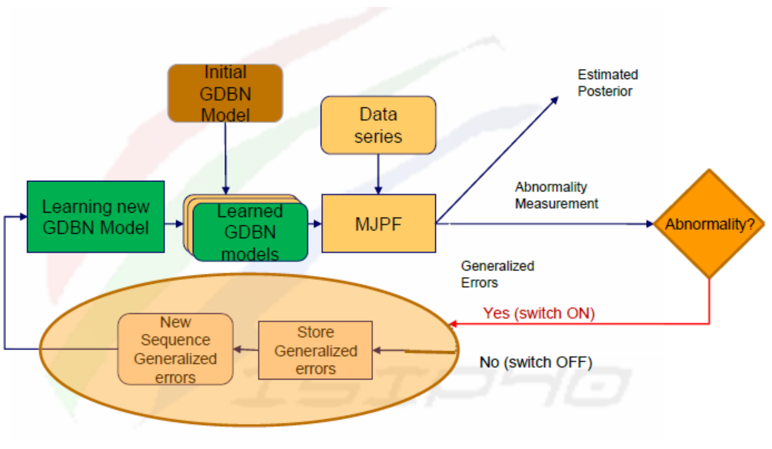


Fig. 27 *Incremental learning of GDBN filters*

In this way, we can generate new models and increase knowledge see Fig. 28 This iterative cycle is done until the measurement parameters for generalized error fall beyond a certain threshold.

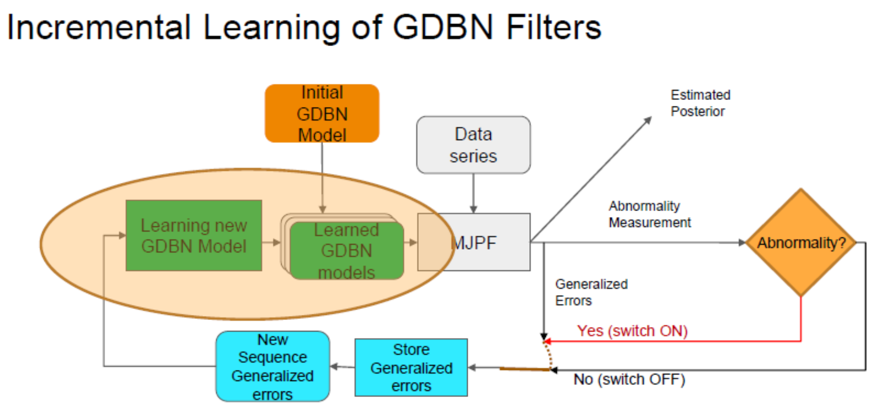


Fig. 28 *Incremental learning of GDBN filters*

4. Conclusion

In this report we presented a method to learn a Markov Jump Particle Filter that estimates the states of moving agents in continuous and discrete level. Our method uses a SOM that codifies regions containing position/velocity information which can be used for detecting and classifying abnormalities.

We trained the MJPF with a training dataset and then tested it in case it had an obstacle (abnormal case). We have obtained three types of measurements to calculate abnormalities in order to increase knowledge.

A possible development of this work would be to make followers interact with the attractors, in the normal and abnormal case.

References

[1] D. C. V. S. L. M. A. C. a. C. R. M. Baydoun, "Learning Switching Models for Abnormality Detection for Autonomous Driving," *2018 21st International Conference on Information Fusion (FUSION)*, pp. 2606-2613, 2018.

[2] T. Kohonen, "Self-Organizing Maps," *ser. Physics and astronomy online library.*, pp. 53-82, 2001.

[3] H. Iqbal, "Clustering Optional for Abnormality Detectionn in Semi-Autonomous Systems.," in *1st international Workshop on Multimodal Understanding and Learning for Embodied Applications*, ACM, 2019.

[4] B. Fritzke, "A growing neural gas network learns topologies," in *Advance in neural information processing systems.*, 1995.

[5] A. G. N. J. &. K. V. Doucet, "Particle filters for state estimation of jump Markov linear systems.," *IEEE Transaction on Signal Processin* , vol. 49(3), no. 10. 1109/78.905890, pp. 613-624, 2001.

[6] "Course Slides".