DITEN

Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture
Polytechnic School, University of Genoa

Practical exam of Cognitive Data Fusion

Jorge Leonardo Quimi Villon

24/04/2020

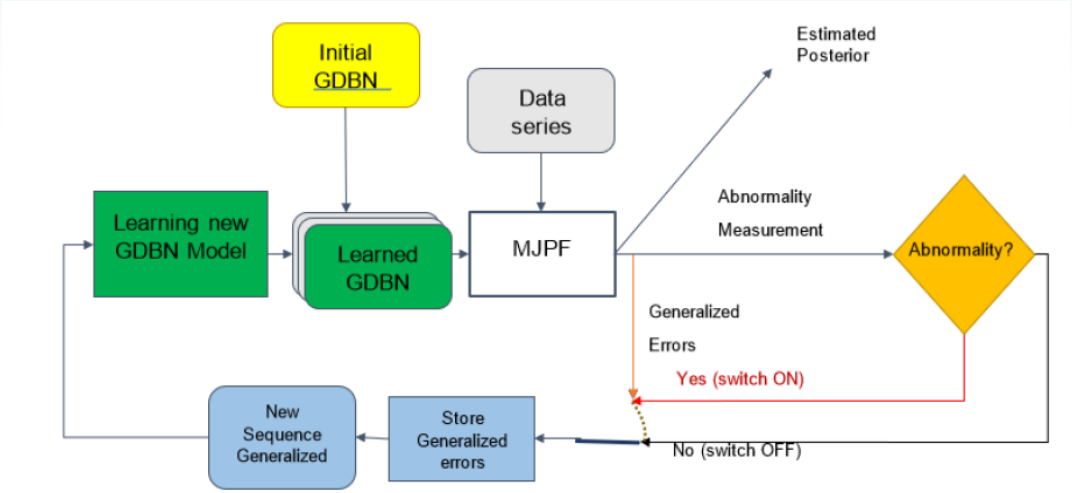
Introduction

- 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 [1]
  - Identification of temporal patterns [2]
  - Autonomous vehicle [2]
- Several state estimation methods have been employed for modeling linear and nonlinear dynamic.
  - Kalman Filter KF
  - Particle Filter PF
  - Markov Jump Particle Filter $MJPF$

[1] C. A. H. &. A.-L. B. Marta C. González, "Understanding individual human mobility patterns," in Nature, vol. 453, June 2008, pp. 779-782.
[2] M. B. L. M. A. C. a. C. S. R. D. Campo, "Modeling and classification of trajectories based on a Gaussian process decomposition into discrete components," in 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, Aug 2017.

Introduction

- In particular, we will focus on detecting abnormalities [1], Through an incremental learning process:

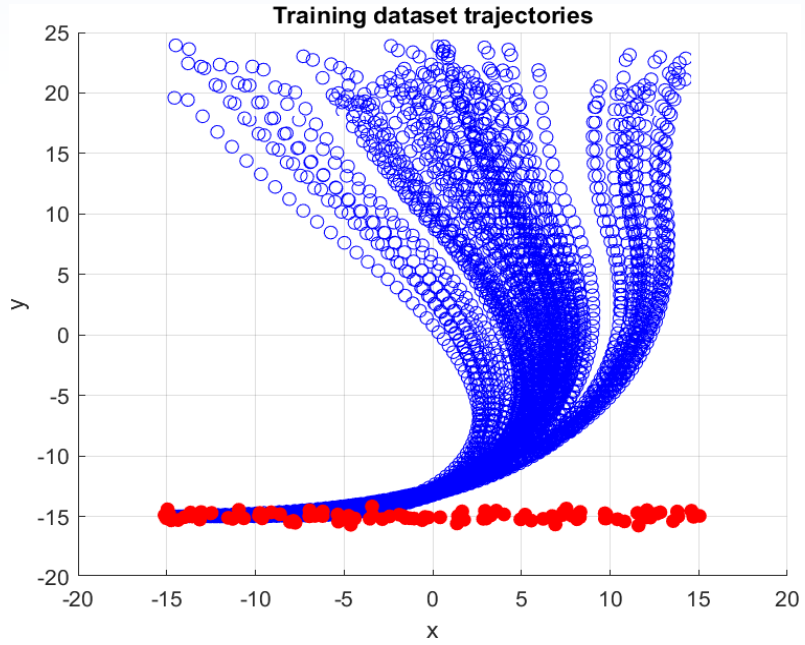


- The sequence of actions can be learned from training data grouped into discrete motion patterns. Abnormalities can be defined as observations that do not match with learned action patterns and models for abnormality detection are generally trained on a set of observations of normal activities.

[1] A. B. L. M. a. C. R. D. Campo, "Static force field representation of environments based on agents' nonlinear motions," in EURASIP Journal on Advances in SIgnal Processing, 1 ed., vol. 2017, Dec 2017, p. 13.

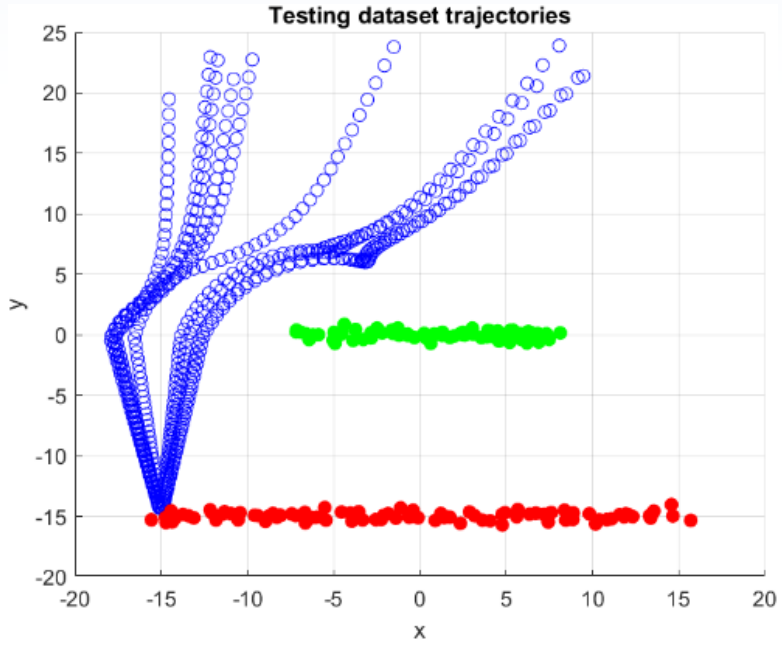
Dataset

- In this project, a dataset describing trajectories is given (in terms of positional data) of two entities that are interacting:



Training dataset (normal)

- Follower trajectories (blue dot)
- Attractor trajectory (red dot)



Testing dataset (abnormal)

- Follower trajectories (blue dot)
- Attractor trajectory (red dot)
- Obstacle trajectory (green dot)

Bayesian Filtering

- Bayesian Filtering is a successful approach to the problem of sequential estimation of the state of a dynamic system by using a sequence of noisy measurement:
- In order to analyze and make inference about a dynamic system usually two model are required:
  - System or dynamic model: it describes the evolution of the state with the time

$$\boldsymbol{X_k} = \boldsymbol{f_{k-1}}(\boldsymbol{X_{k-1}}, \boldsymbol{n_{k-1}}) \qquad \boldsymbol{x_k} \in \mathbb{R}^{\boldsymbol{N}}$$

  - Measurement model: it relates the noisy measurements with to the state:

$$\boldsymbol{Z_k} = \boldsymbol{h_k}(\boldsymbol{X_k}, \boldsymbol{W_k}) \qquad \boldsymbol{Z_k} \in \mathbb{R}^{\boldsymbol{M}}$$

- Recursive filtering
- Two basic stages are involved in the filtering process:
  - Prediction: $p(\boldsymbol{X_k}|\boldsymbol{Z_{k-1}}) = \int p(\boldsymbol{X_k}|\boldsymbol{X_{k-1}}) p(\boldsymbol{X_{k-1}}|\boldsymbol{Z_{k-1}}) d\boldsymbol{X_{k-1}}$
  - Update: $p(\boldsymbol{X_k}|\boldsymbol{Z_k}) = p(\boldsymbol{X_k}|\boldsymbol{Z_k}, \boldsymbol{Z_{k-1}}) = \frac{p(\boldsymbol{Z_k}|\boldsymbol{X_k}, \boldsymbol{Z_{k-1}}) p(\boldsymbol{X_k}|\boldsymbol{Z_{k-1}})}{p(\boldsymbol{Z_k}|\boldsymbol{Z_{k-1}})} = \frac{p(\boldsymbol{Z_k}|\boldsymbol{X_k}) p(\boldsymbol{X_k}|\boldsymbol{Z_{k-1}})}{p(\boldsymbol{Z_k}|\boldsymbol{Z_{k-1}})}$

Kalman Filter

- Optimal finite-dimensional al algorithm for recursive Bayesian state estimation are available in linear-Gaussian cases, provided by the Kalma Filter, and in few other more specific and not common situations.
- The Kalman filter assumes that posterior pdf at every step is Gaussian and then it can be completely characterized by the mean and the covariance.
- These assumptions hold if $\boldsymbol{n_{k-1}}$ and $\boldsymbol{W_k}$ are drawn from Gaussian density and the functions $\boldsymbol{f_{k-1}}$ and $\boldsymbol{h_k}$ are linear. Consequently:

$$\begin{aligned} \boldsymbol{X_k} &= \boldsymbol{f_{k-1}}(\boldsymbol{X_{k-1}}, \boldsymbol{n_{k-1}}) \\ \boldsymbol{Z_k} &= \boldsymbol{h_k}(\boldsymbol{X_k}, \boldsymbol{W_k}) \end{aligned} \quad \Rightarrow \quad \begin{aligned} \boldsymbol{X_k} &= \boldsymbol{F_{k-1}}\boldsymbol{X_{k-1}} + \boldsymbol{n_{k-1}} \\ \boldsymbol{Z_k} &= \boldsymbol{H_k}\boldsymbol{X_k} + \boldsymbol{W_k} \end{aligned}$$

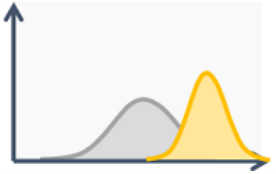
- The matrices $\boldsymbol{F_{k-1}}$ (of dimensional $n_x \times n_x$) and $\boldsymbol{H_k}$ (of dimension $n_z \times n_x$) define the linear functions.
- Random sequences $\boldsymbol{n_{k-1}}$ and $\boldsymbol{W_k}$ are mutually independent zero-mean white Gaussian with covariance Q_{k-1} and R_k respectively.

Kalman Filter

Prediction

Update

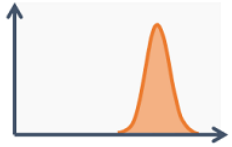
Z_k



$$v_k = Z_k - H_k\hat{X}_{k|k-1}$$
$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

$$\hat{X}_{k|k-1} = F_{k-1}\hat{X}_{k-1|k-1}$$
$$P_{k|k-1} = Q_{k-1} + F_{k-1}P_{k-1|k-1}F_{k-1}^T$$

$$K_k = P_{k|k-1}H_k^T S_k^{-1}$$



$$\hat{X}_{k|k} = \hat{X}_{k|k-1} + K_k\left(Z_k - H_k\hat{X}_{k|k-1}\right)$$
$$P_{k|k} = [I - K_k H_k]P_{k|k-1}$$

$k+1$

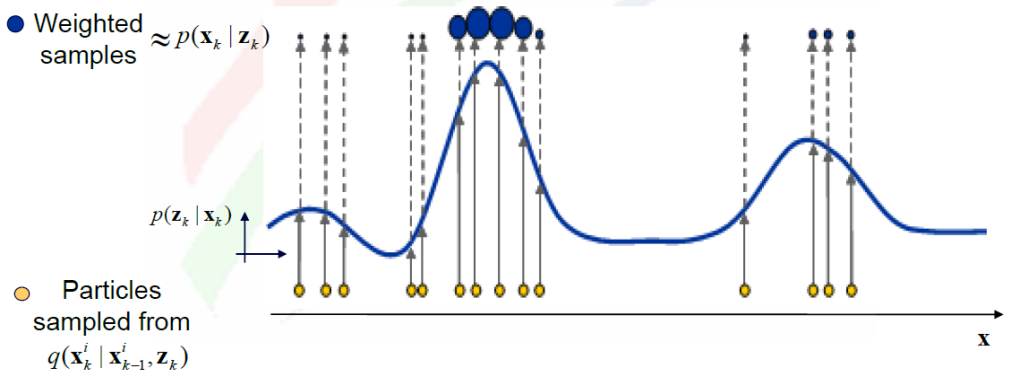
$\hat{X}_{k|k}$

Particle filter

- 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.
- Set of N particles (x_k^i, w_k^i) are used to represent posteriors $\boldsymbol{BEL(X_k)}$
- **Sequential Importance Sampling** (SIS) algorithm is a Monte Carlo Integration method where importance sampling is used to perform non-linear filtering.

The **posterior is approximated** as:

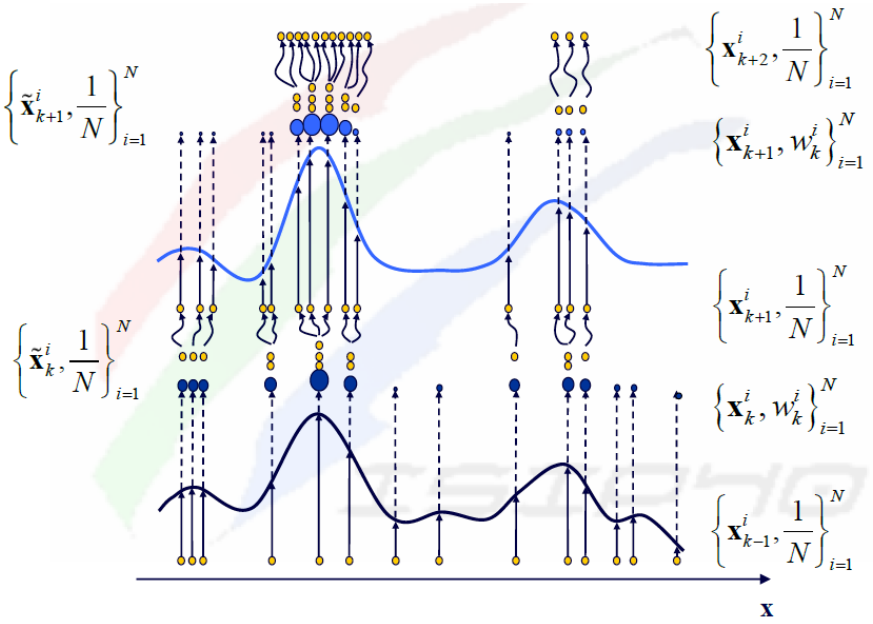
$$p(\mathbf{x}_k \mid \mathbf{z}_k) \approx \sum_{i=1}^{N} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i)$$



- **Degeneracy problem**: consisting in the progressive annihilation of weight update formula. After several recursive steps all particles will tend to have *negligible weights*.

Particle filter

- To avoid this problem **resampling** is used to redistribute particles weight by renormalizing weights at each step.
  - Sample with low weights are **eliminated**
  - Particles with high weights are **multiplied**
- In general the idea is to redistribute the particles in a proper way givin to each of them the same weight
- **Sequential Importance Resampling** (SIR)



Markov Jump Particle Filter

- 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.
- Contextually linear models with Gaussian noise [1]:

$$X_k = F_{S_{k-1}} X_{k-1} + n_k$$
$$Z_k = H_{S_{k-1}} X_k + m_k$$

Different Dynamic models based on switching variable.

Different Observation models based on switching variable.

$$p(X_{0:k}, S_{1:k}|Z_{1:k}) = p(X_{0:k}|S_{1:k}, Z_{1:k}) p(S_{1:k}|Z_{1:k})$$

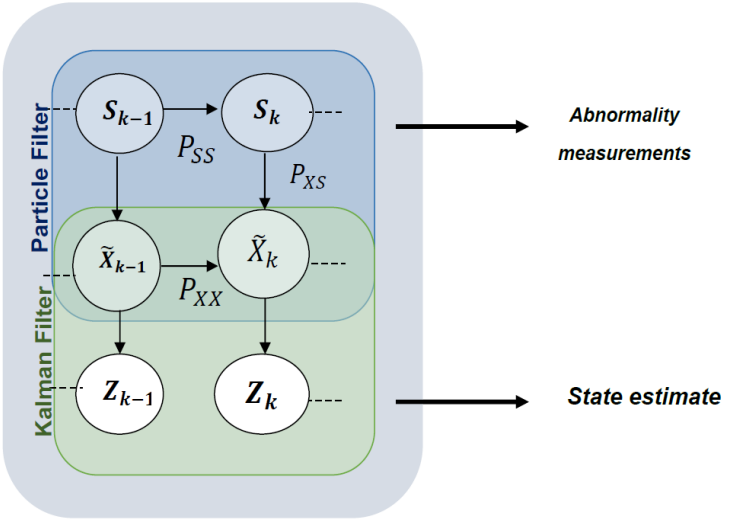
Gaussian posterior model when switching variable is fixed: estimate this with Kalman Filter

Posterior of the discrete switching variable: particle Filter can be used for related non linear dynamic model

[1] 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.

Markov Jump Particle Filter

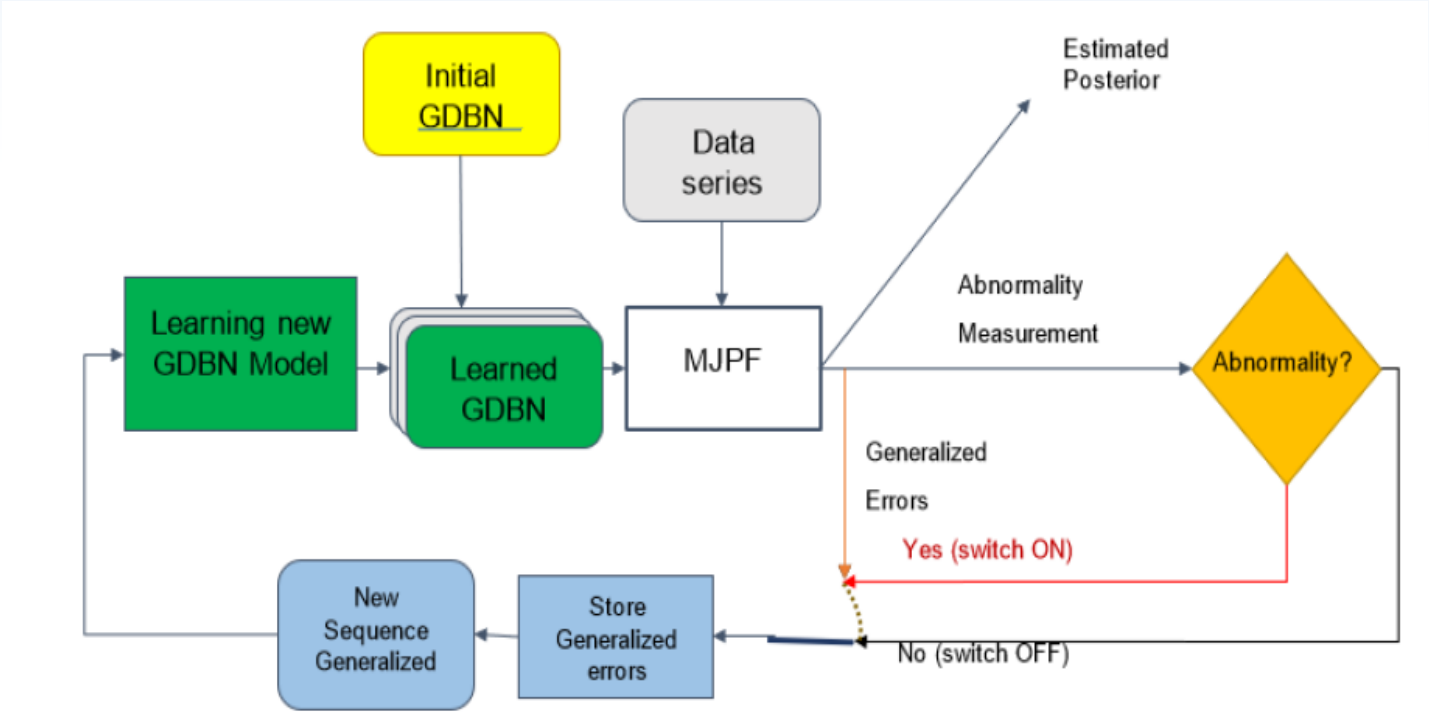
- GDBN for MJPF is shown below, this represents a hybrid model: continuous and discrete hidden variables
  - Continuous variables have conditional linear Gaussian dynamic
  - Discrete variable has no continuous parameters



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

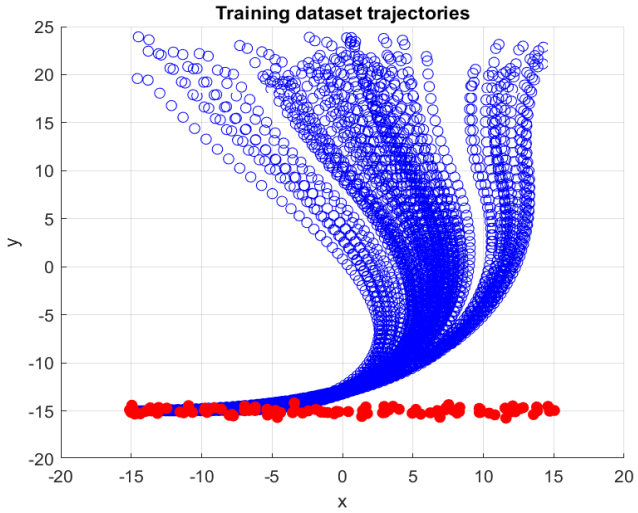
- $\tilde{X}_k$ is made up of position and velocity of the object and constitute the continues level, approximated by a Kalman Filter.
- The superstate S_k is a discrete variable that is approximated by a Particle Filter

Incremental Learning phases

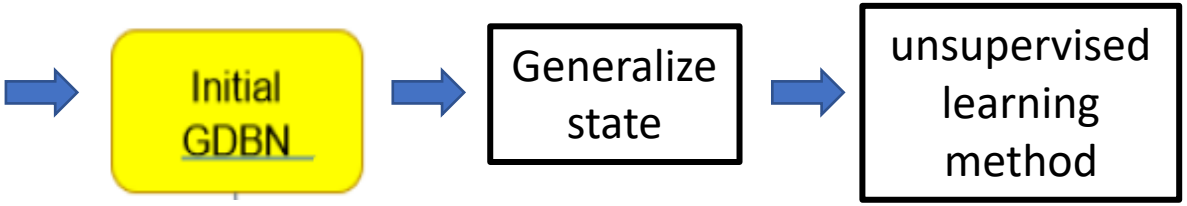


Training phase (NFF)

- The **initial $GDBN$** is created through a Null-force filter (um-motivate Kalman filter)
- Un-Motivated Kalman Filter $(UMKF)$ is a Kalman filter under the simplest situation, in which the object is influenced by NO forces. In such a case, the object should keep staying in the same position with no movement. In this case, for a moving object, the error/innovation output from the filter could be the velocity of the object.
- Null Force Filter produce a **generalized state** containing filtered position of the object and the error/innovation.



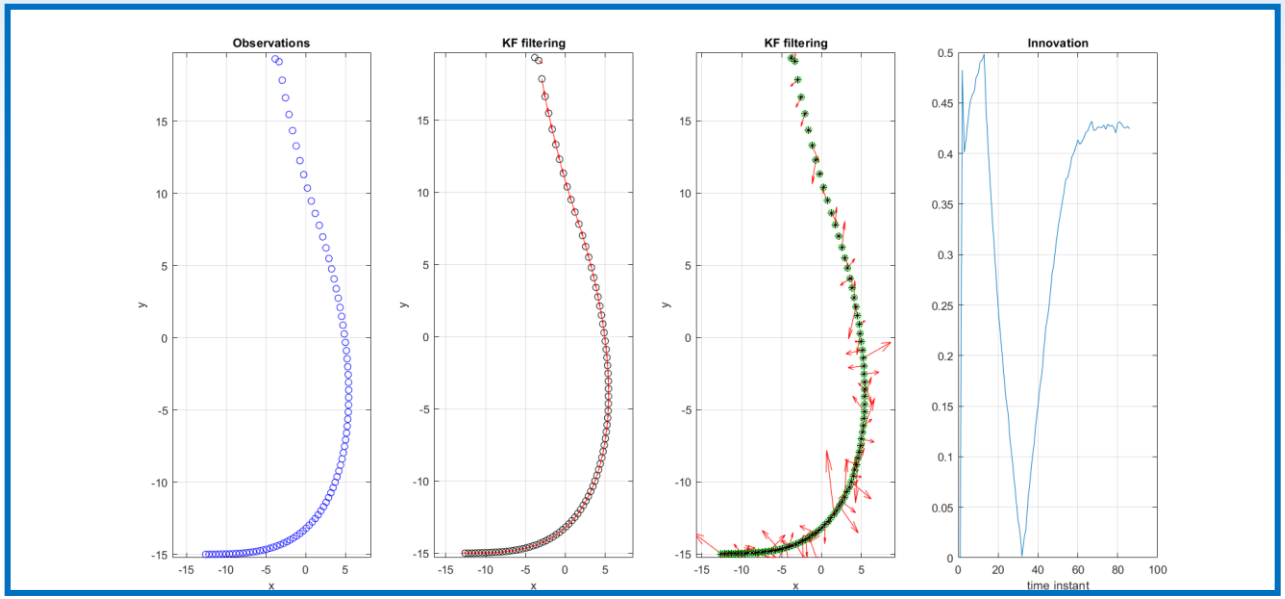
NFF(UMKF)



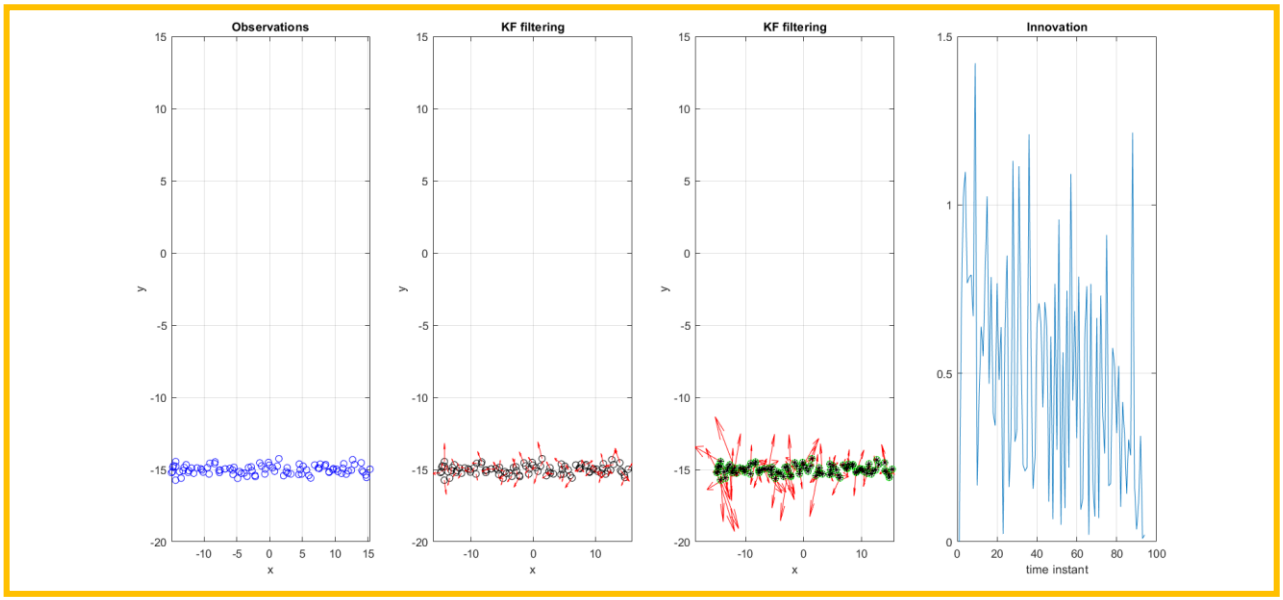
Follower trajectories (blue dots)
Attractor trajectory (red dots)

Incremental Learning (UMKF)

Follower

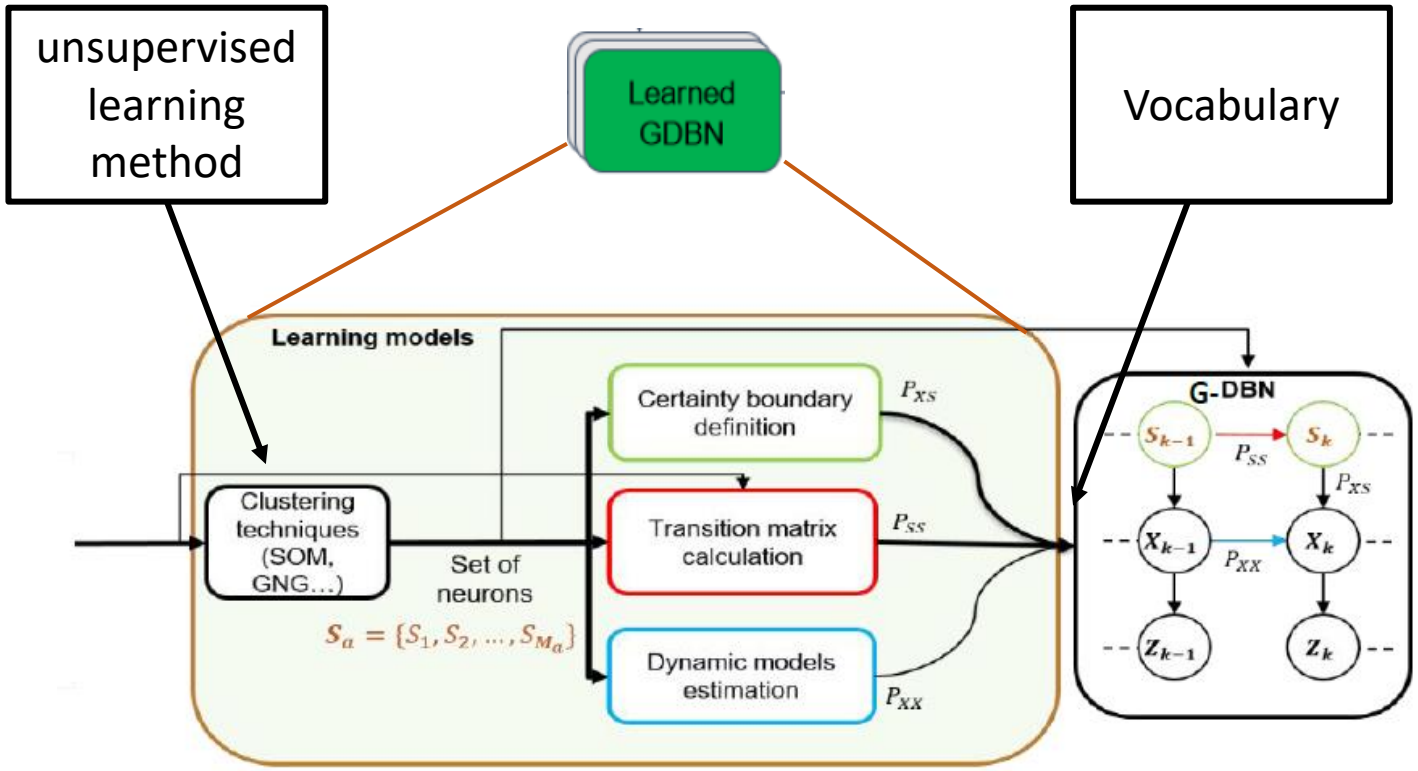


Attractor



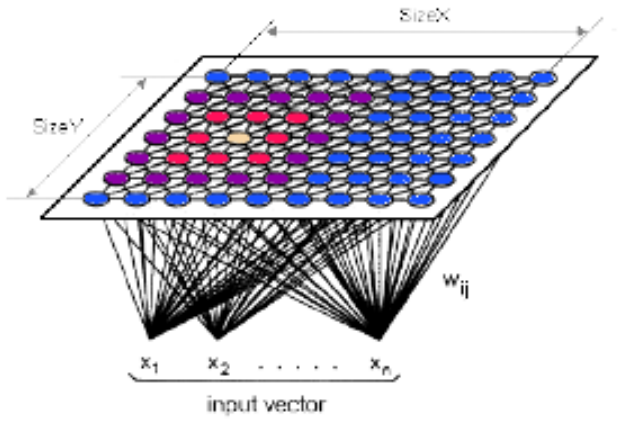
Unsupervised clustering

- The next step is to learn Generalized Dynamic Bayesian Network (GDBN)
- We want to learn vocabularies to be used at the discrete level within a hierarchical DBN
- Let us look at machine learning algorithm for unsupervised clustering

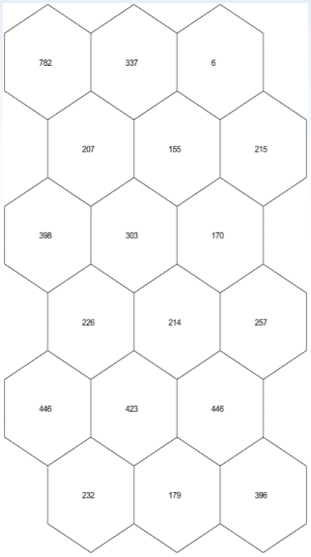


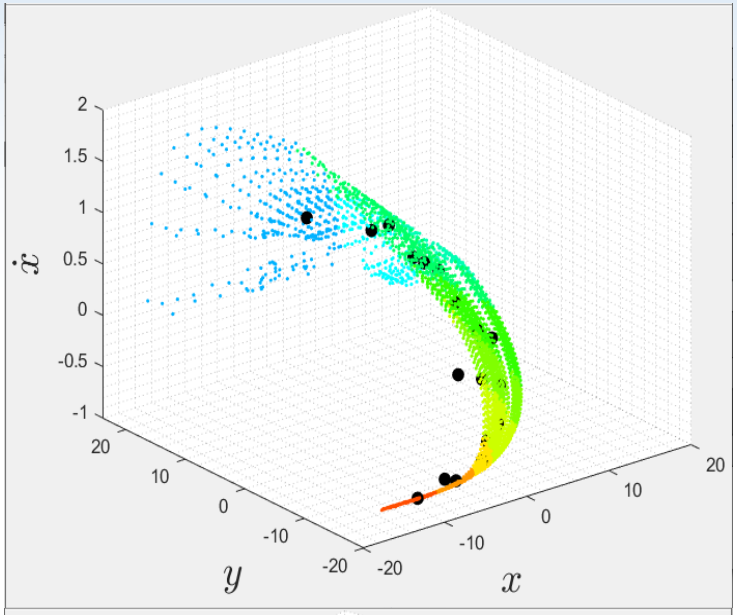
Self Organizing Map (SOM)

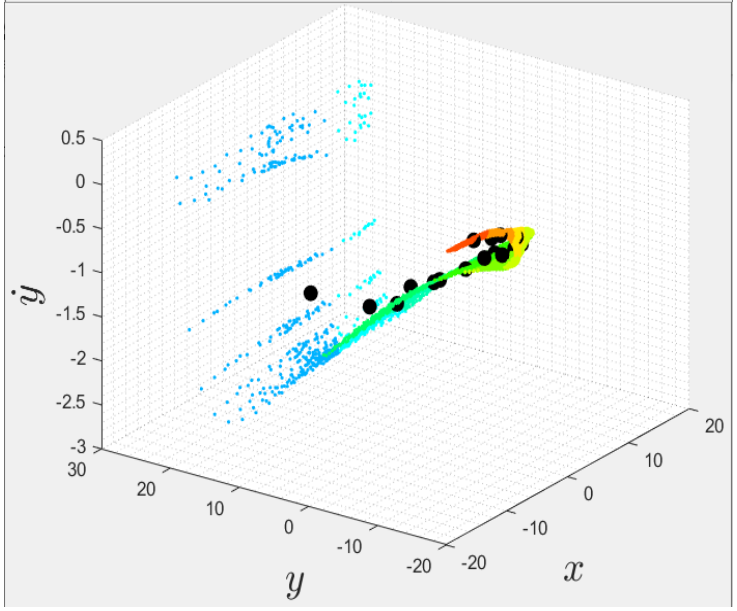
- The self-organizing map (SOM) is mainly a method for unsupervised learning, based on a grid of artificial neurons whose weights are adapted to match input vectors in a training set.
- The SOM is an unsupervised Neural network technique that approximates an unlimited number of input data by a finite set of models arranged in a 2d (rarely 3d) grid, where neighbor nodes correspond to more similar models.
- The models are produced by a learning algorithm that automatically orders them on the two-dimensional grid along with their mutual similarity.

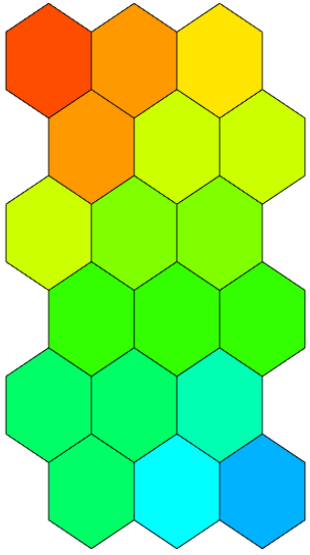


(SOM) Followers trajectories

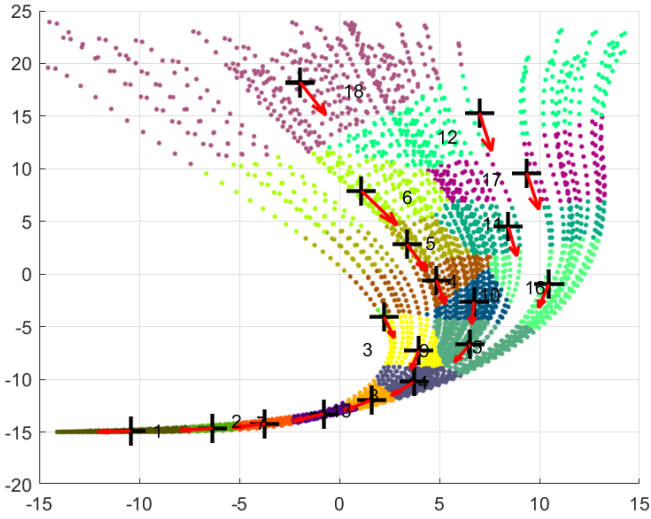




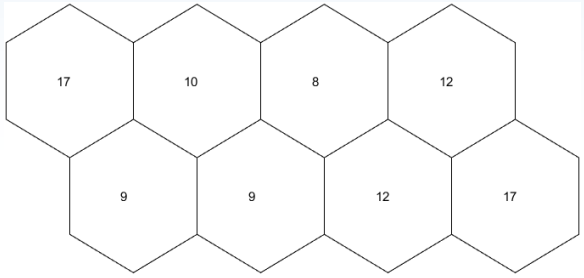


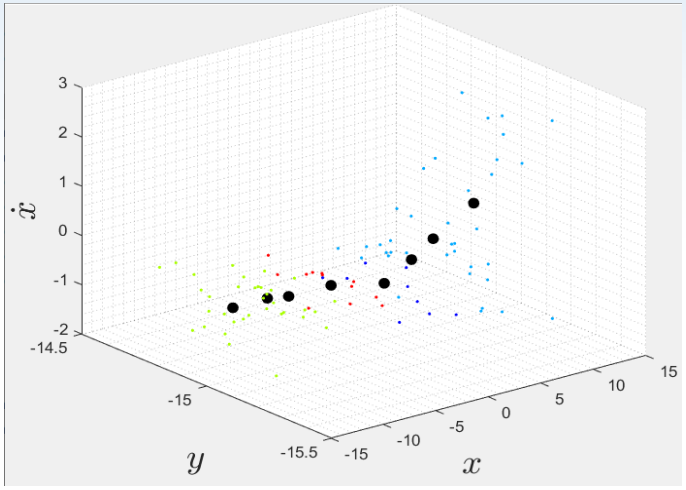


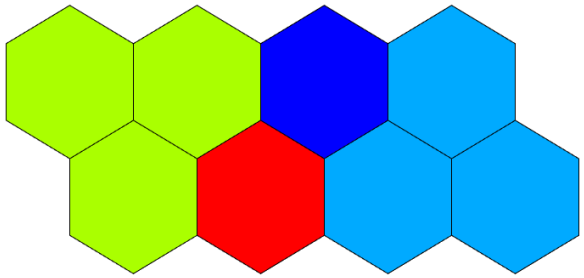
Vocabulary

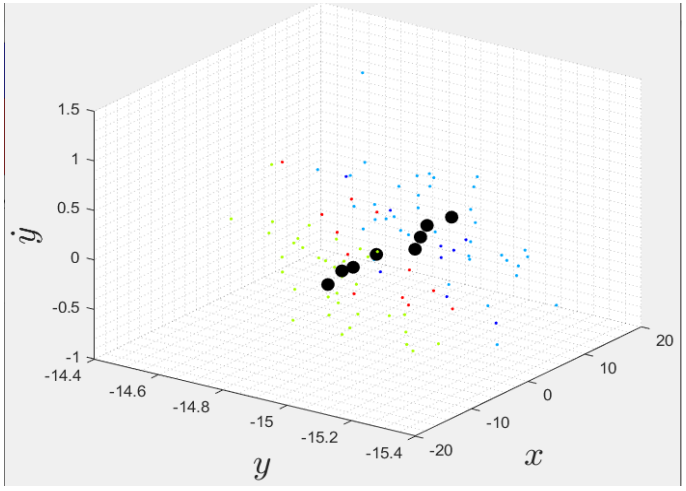


(SOM) Attractor trajectory

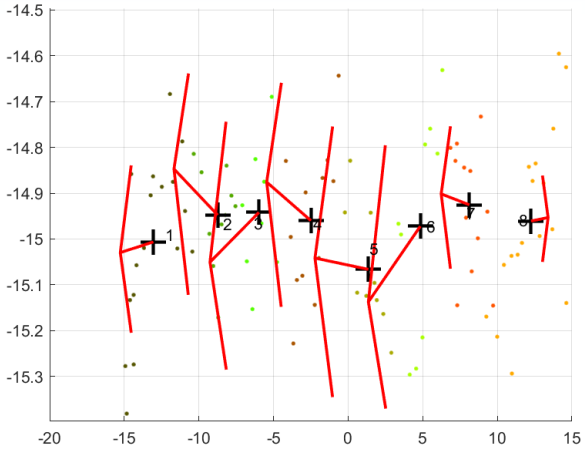






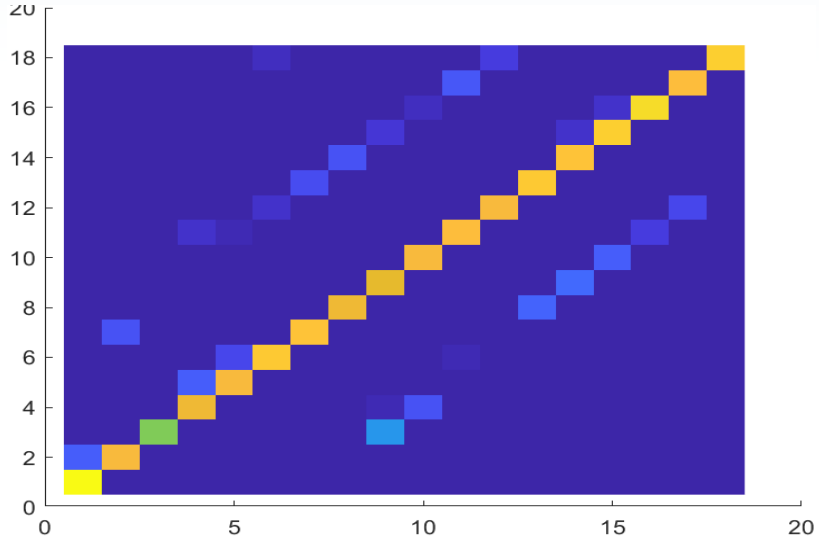


Vocabulary

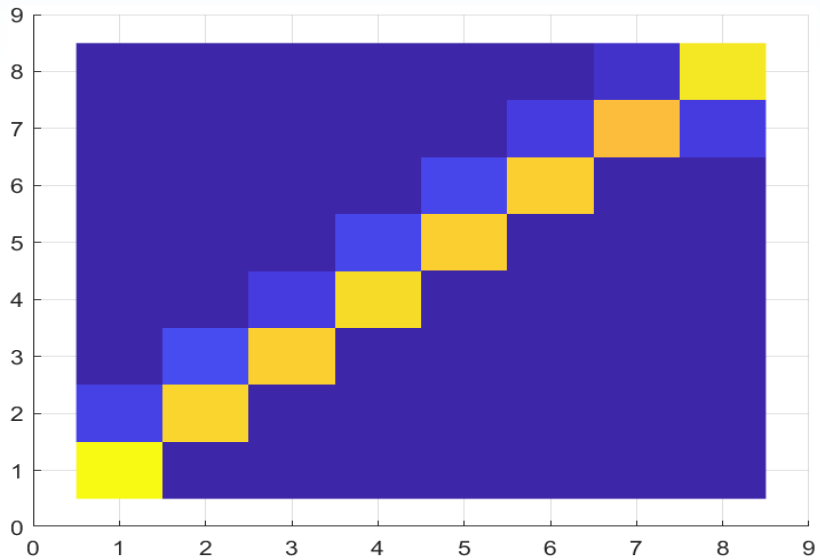


Transition Matrix

- 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.



Followers

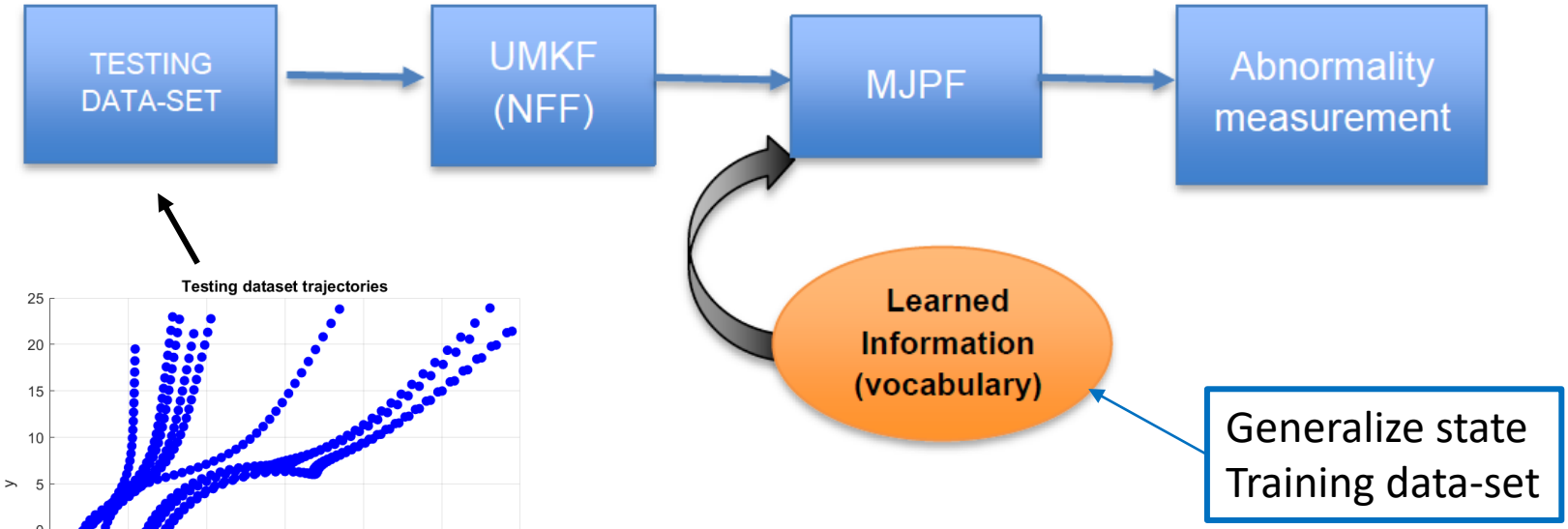


Attractor

- Yellow means that the probability of staying in the same cluster is high in the next step
- Intense blue means that the probability of switching from one cluster to another is low

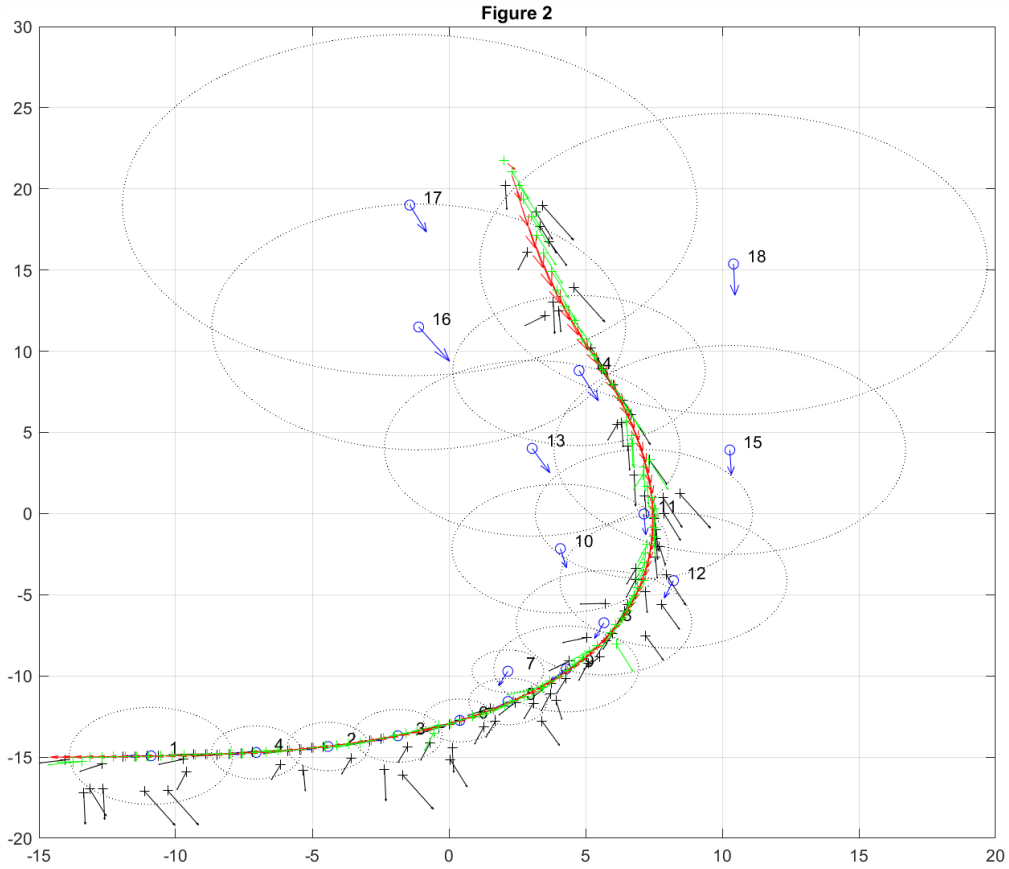
Testing phase (NFF)

- Also in this case (as in the training phase) the test trajectory dataset must be generalized.
- Null Force Filter (un-motivated Kalman filter) produce a generalized state containing filtered position of the object and the error/innovation.



$MJPF$ (training)

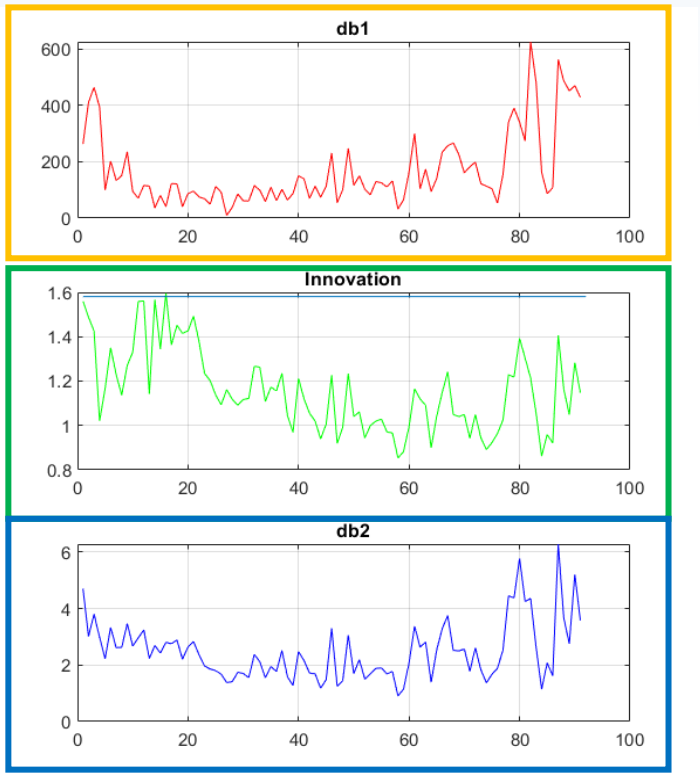
- The $MJPF$ is initially trained with the vocabulary from the training phase
- In this way the MJPF will have dynamic models on which to make the comparison, when there will be a different trajectory at the entrance



Blue circle: mean position of cluster
Blue arrow: mean velocity vector of cluster
Red arrows: velocity vectors of training data
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 training data

$MJPF$ (training)

- In this case, three types of measurement are used:



$db1$ measure the Bhattacharyya distance between the distribution of the prediction and the likelihood of the discrete state.

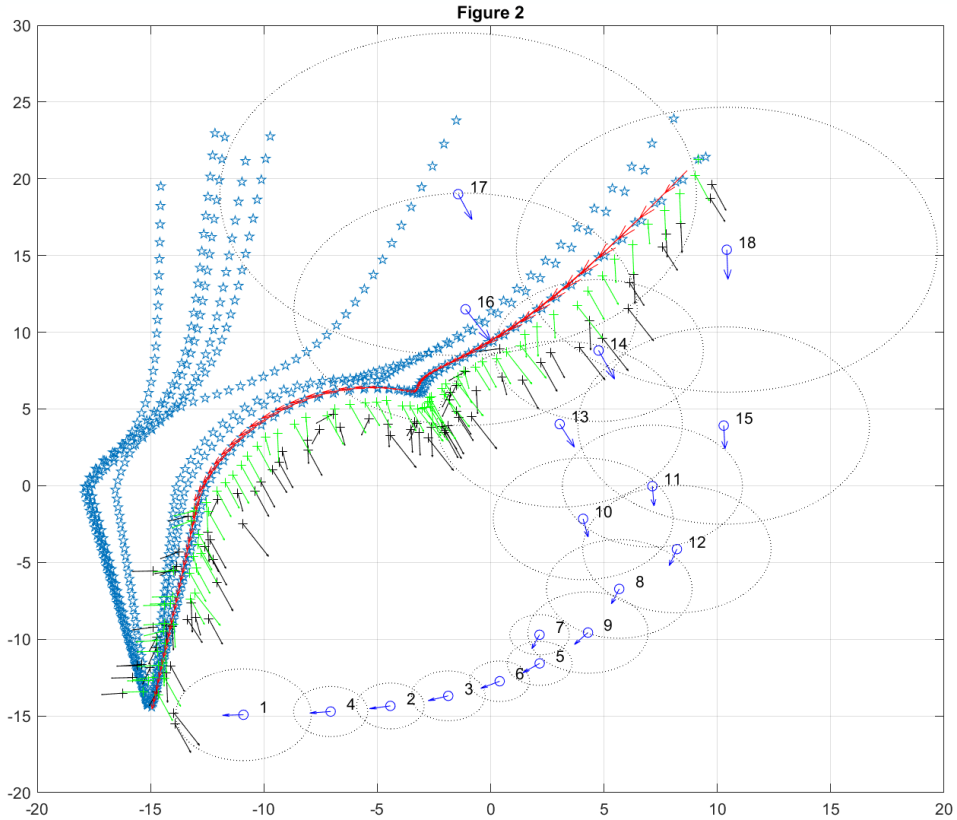
Innovation, this value was computed as a simple Euclidean distance between prediction and observation

$db2$ measures the Bhattacharyya distance between the distribution of the prediction and the likelihood of the continuous state.

- The **threshold** in innovation allows us to classify abnormalities in the testing phase of the MJPF

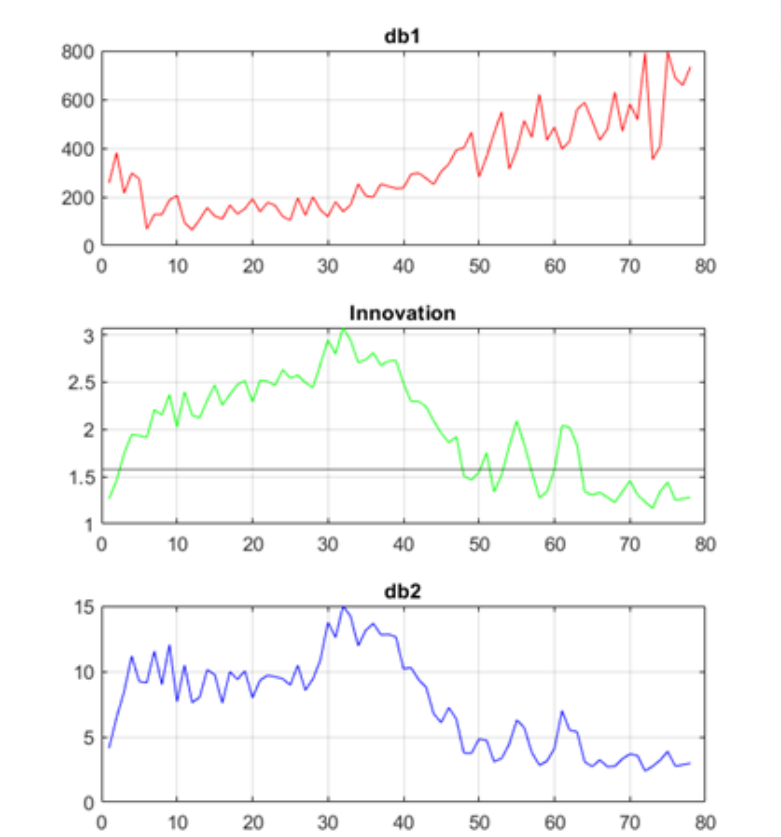
$MJPF$ (testing)

- In this case, 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.



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

$MJPF$ (testing)

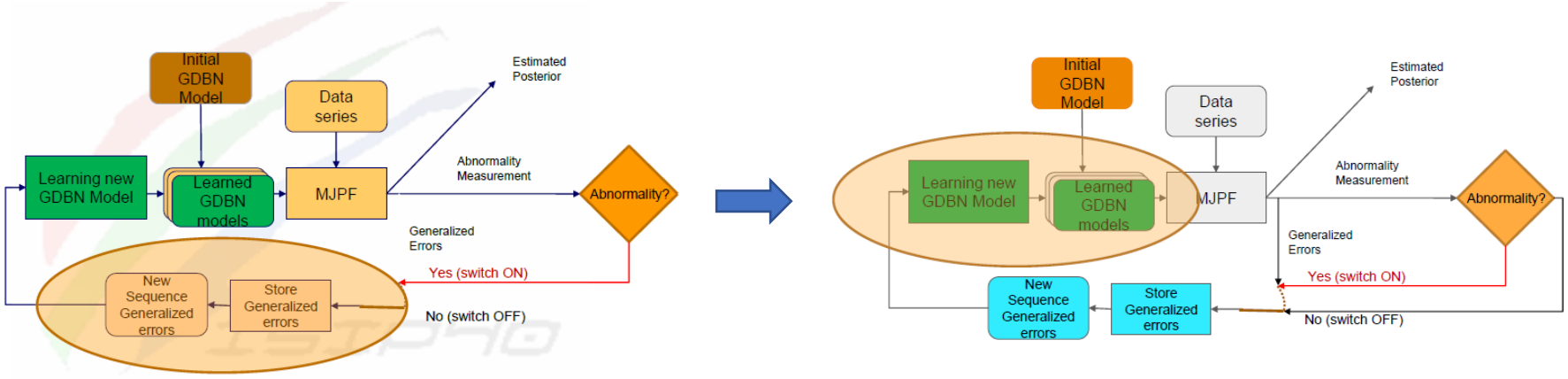


In order to distinguish in an automatic way abnormal scenario, we need to identify a threshold (obtained in the training phase) for the innovation.

- 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.

Abnormalities detect

- These abnormalities can be saved to generate new sequences based on these generalized errors
- In this way, we can generate new models and increase knowledge



Conclusion

- Method to learn a Markov Jump Particle Filter that estimates the states of moving agents in continuous and discrete level.
- SOM
- A further development of this study could be the iteration between the attractors and the followers, in normal and abnormal case (when there is an obstacle).