# Incremental Learning Of Evolving Fuzzy Inference Systems : Application To Handwritten Gesture Recognition

Abdullah Almaksour

THÈSE INSA Rennes

*sous le sceau de l'Université Européenne de Bretagne*

pour obtenir le grade de

DOCTEUR DE L'INSA DE RENNES

*Spécialité : Informatique*


présentée par

**Abdullah Almousa Almaksour**

ÉCOLE DOCTORALE : MATISSE

LABORATOIRE : IRISA – UMR6074


# Incremental Learning Of Evolving Fuzzy Inference Systems : Application To Handwritten Gesture Recognition

**Thèse soutenue le 29 juillet 2011**

devant le jury composé de :

**M. Mohamed Cheriet**
Professeur à l'ETS de Montréal / *Rapporteur*

**M. Lionel Prevost**
Professeur à l'Université des Antilles / *Rapporteur*

**M. Réjean Plamondon**
Professeur à L'Ecole Polytechnique de Montréal / *Examinateur*

**M. Christian Viard-Gaudin**
Professeur à L'Université de Nantes / *Examinateur*

**M. Moamar Sayed-Mouchaweh**
Maitre de conférences à l'Université de Reims / *Examinateur*

**M. Eric Anquetil**
Professeur à l'INSA de Rennes / *Directeur de thèse*

# Contents

**1 General Introduction** **5**

**2 Evolving Classification Systems: State Of The Art** **14**

2.1 Preamble . . . . . 15

2.2 Definitions . . . . . 15

2.2.1 Batch learning vs. Incremental learning . . . . . 16

2.2.2 Offline learning vs. Online learning . . . . . 17

2.2.3 Active learning vs. Incremental learning . . . . . 17

2.2.4 Instance memory vs. Concept memory . . . . . 18

2.2.5 Incremental learning criteria . . . . . 19

2.2.6 Discussion . . . . . 20

2.3 Adaptive classification systems . . . . . 21

2.3.1 AdaBoost-based learning . . . . . 22

2.3.2 Dynamic Time Warping . . . . . 23

2.3.3 Adaptive Neuro-Fuzzy Inference Systems (ANFIS) . . . . . 24

2.3.4 Adaptation by Adjustment of Prototypes (ADAPT) . . . . . 26

2.3.5 Discussion . . . . . 27

2.4 Incremental clustering . . . . . 28

2.4.1 Distance-based Incremental clustering . . . . . 28

2.4.1.1 Adaptive Resonance Theory (ART) Networks . . . . 29

2.4.1.2 Vector Quantization . . . . . 29

2.4.1.3 Evolving Clustering Method (ECM) . . . . . 30

2.4.2 Density-based Incremental clustering . . . . . 31

2.4.2.1 Graph-based Incremental clustering methods . . . . 31

2.4.2.2 Recursive Density Estimation (eClustering) . . . . . 32

2.4.3 Discussion . . . . . 33

2.5 Overview of some evolving systems . . . . . 34
2.5.1 Evolving Ensemble Systems . . . . . 35
2.5.1.1 Learn++ . . . . . 35
2.5.1.2 Growing Negative Correlation Learning . . . . . 36
2.5.2 Fuzzy inference systems . . . . . 37
2.5.2.1 Fuzzy set theory . . . . . 37
2.5.2.2 Fuzzy inference systems structures . . . . . 38
2.5.2.3 Takagi-Sugeno FIS . . . . . 40
2.5.2.3.1 Single vs. multiple output . . . . . 41
2.5.2.3.2 Antecedent structure . . . . . 41
2.5.2.3.3 Inference process . . . . . 43
2.5.2.3.4 Consequence variants . . . . . 44
2.5.3 Evolving Fuzzy inference systems . . . . . 47
2.5.3.1 FLEXible Fuzzy Inference System (FLEXFIS) . . . . 47
2.5.3.2 Evolving Fuzzy Neural Networks (EFuNN) . . . . . 48
2.5.3.3 Generalized Fuzzy MIN-MAX (GFMM) . . . . . 49
2.5.3.4 Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) . . . . . 50
2.5.3.5 Evolving Takagi-Sugeno (eTS) . . . . . 51
2.5.4 Discussion . . . . . 52
2.6 Incremental Feature selection . . . . . 56
2.6.1 Incremental Principal component analysis . . . . . 56
2.6.2 Incremental linear discriminant analysis . . . . . 57
2.7 Conclusion . . . . . 58

**3 Evolve(+)(+): incremental learning of evolving Takagi-Sugeno classifiers with enhanced antecedent structure 60**
3.1 Introduction . . . . . 61
3.2 System architecture . . . . . 63
3.3 Evolve: an incremental learning algorithm for evolving TS classifiers . 66
3.3.1 Density-based incremental clustering . . . . . 68
3.3.2 Antecedent adaptation . . . . . 71
3.3.3 Consequent learning . . . . . 73
3.3.3.1 Global learning of consequent parameters . . . . . 73
3.3.3.2 Local learning of consequent parameters . . . . . 76

3.3.3.3 Learning of zero-order consequences . . . . . 77
3.3.4 Learning stability using delayed antecedent adaptation . . . . 78
3.3.5 Evolve: the complet algorithm . . . . . 79
3.4 A novel paradigm for stable incremental learning of TS models . . . 81
3.4.1 Evolve+: antecedent learning based on global error feedback . 83
3.4.2 Evolve++: confusion-driven antecedent learning . . . . . 85
3.5 Open issues . . . . . 87
3.5.1 Forgetting strategy . . . . . 87
3.5.2 Ordering effect . . . . . 89
3.6 Conclusion . . . . . 90

**4 Experimental Validation** **93**
4.1 Introduction . . . . . 94
4.2 Classification datasets . . . . . 95
4.2.1 SIGN dataset . . . . . 95
4.2.2 UCI datasets . . . . . 96
4.3 Experimental protocol . . . . . 97
4.4 Experimental results . . . . . 100
4.4.1 Global evaluation of the different learning algorithms . . . . . 100
4.4.2 Results for different consequent structures and learnings strategies . . . . . 107
4.4.3 Correction efforts required by the different algorithms . . . . . 111
4.4.4 Performance evaluation for unsynchronized incoming classes . 117
4.5 Real applications . . . . . 120
4.6 Conclusion . . . . . 122

**5 Synthetic Data Generation for Evolving Handwriting Classifiers 125**
5.1 Introduction . . . . . 126
5.2 Related works . . . . . 128
5.2.1 Class-dependent geometric deformations . . . . . 128
5.2.2 Class-independent geometric deformations . . . . . 131
5.2.3 Motor-based deformations . . . . . 134
5.3 Sigma-Lognormal model . . . . . 135
5.4 Synthetic handwritten gesture generation using Sigma-Lognormal model 139

5.5 Evolving handwriting classifier learning acceleration using synthetic data . . . 141
5.6 Experimental results . . . 142
5.7 Conclusion . . . 146

**6 General Conclusion and Future Work** **148**
6.1 Conclusion . . . 149
6.2 Future works . . . 151

# List of Figures


1.1 The life cycle of static classifiers with distinct learning and operation phases . . . . . 7
1.2 Simultaneous operation and learning (incremental) processes in evolving classification systems . . . . . 8

2.1 The point-to-point correspondence of two characters established with a DTW-algorithm [1] . . . . . 24
2.2 Functional equivalence between a traditional fuzzy inference system and ANFIS [2] . . . . . 25
2.3 Principle of the ADAPT compromises between the optimizations of all classes [3] . . . . . 27
2.4 Basic ART structure [4] . . . . . 30
2.5 ECM process with consecutive examples $x_1$ to $x_9$ [5] . . . . . 31
2.6 The relationship between the sparse graph and the representative sparse graph [6] . . . . . 32
2.7 The difference between crisp and fuzzy sets . . . . . 37
2.8 Different membership functions: (a) Triangular, (b) Trapezoidal, and (c) Gaussian. . . . . 38
2.9 The different steps of the fuzzy inference mechanism for a given toy example . . . . . 39
2.10 (a) Hyper-rectangle zones of influence obtained by the aggregation of triangular membership functions [5], (b) hyper-spherical (or elliptical) zones of influence obtained by the aggregation of Gaussian membership functions [7]. . . . . 42
2.11 Zero-order TS model presented in the form of RBF network . . . . . 45
2.12 First-order TS model presented as a neural network . . . . . 46

2.13 A rule in EFuNN represents an association of two hyper-spheres from the fuzzy input space and the fuzzy output space. . . . . 48
2.14 The three-layer neural network that implements the GFMM algorithm [8] . . . . . 50

3.1 The zone of influence of the prototype is parallel to the axes in (a), (c) and (e), while the rotated zones in (b), (d) and (f) result in more accurate data covering. . . . . 65
3.2 The different components of Evolve algorithm . . . . . 68
3.3 An example to illustrate the concept of the potential of data points [9]. . . . . 69
3.4 The new learning paradigm: antecedent adaptation driven by output feedback . . . . . 83
3.5 The difference between stationary and non-stationary averaging. . . . 88
3.6 Order-independent incremental learning system [10] . . . . . 90

4.1 Handwritten gestures in the dataset SIGN . . . . . 96
4.2 Learning/test protocols for incremental learning problems . . . . . 98
4.3 Evolution of classification performance during the incremental learning process (SIGN) . . . . . 101
4.4 (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (Sign dataset) . . . . . 103
4.5 (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (CoverType dataset) . . . . . 104
4.6 (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (PenDigits dataset) . . . . . 105
4.7 (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (Segment dataset) . . . . . 106
4.8 Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (PenDigits dataset) . . . . . 109

4.9 Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (CoverType dataset) . . . . . 110
4.10 Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Segment dataset) . . . . . 110
4.11 Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Letters dataset) . . . . . 110
4.12 Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Sign dataset) . . . . . 111
4.13 Accumulated number of misclassification errors committed on the learning samples (SIGN) . . . . . 113
4.14 Accumulated number of misclassification errors committed on the learning samples (CoverType) . . . . . 113
4.15 Accumulated number of misclassification errors committed on the learning samples (PenDigits) . . . . . 114
4.16 Accumulated number of misclassification errors committed on the learning samples (Segment) . . . . . 114
4.17 Evolution of misclassification rate according to accumulated number of corrections (SIGN) . . . . . 115
4.18 Evolution of misclassification rate according to accumulated number of corrections (CoverType) . . . . . 116
4.19 Evolution of misclassification rate according to accumulated number of corrections (PenDigits) . . . . . 116
4.20 Evolution of misclassification rate according to accumulated number of corrections (Segment) . . . . . 117
4.21 Performance stability and recovery when introducing new unseen classes (SIGN) . . . . . 119

5.1 Synthetic samples (in bold) generated by deforming the reference pattern according to the first two Eigen-deformations (figure obtained from [11]) . . . . . 129
5.2 Handwriting generation using class-dependent deformation presented in [12]; (a) samples of letter “d” from training data, (b) synthesized “d” generated using trained deformation models, (c) samples of “s” from training data, and (d) synthesized “s” generated using trained deformation models . . . . . 130

5.3 Example of class-dependent geometric deformation, the deformation observed in (a) for the letter “M” is rarely observed for the letter “H” as illustrated in (b) (figure obtained from [13]) . . . . . 130
5.4 Example of class-independent geometric deformation used [14]. Small random scaling and rotation of each stroke piece are applied to each stroke. . . . . 131
5.5 Examples of synthetic characters generated by the four distortion techniques proposed in [15] . . . . . 133
5.6 Example of the reconstruction of handwriting movements using Sigma-Lognormal model (figure obtained from [16]) . . . . . 137
5.7 Some examples of generated gestures . . . . . 140
5.8 Incorporating lognormal-based data generation in the learning of evolving handwriting classifiers . . . . . 141
5.9 The handwritten gestures used in the handwriting generation experiments (a subset of Sign dataset) . . . . . 142
5.10 Performance improvement when using synthetic data generation in the incremental learning process . . . . . 143
5.11 Evaluation of the impact of synthetic samples when adding new gestures 144
5.12 Evaluation of distortion combinations presented in Table 5.1 . . . . . 145

# List of Tables

2.1 Incremental learning methods categories depending on their used memories . . . . . 19
2.2 Learning steps in ANFIS . . . . . 25
2.3 The main characteristics of existing evolving FISs . . . . . 53

3.1 A case study to illustrate the difference in weighting strategies between Evolve+ and Evolve++ . . . . . 87

4.1 Characteristics of learning datasets . . . . . 97
4.2 The size of the learning subsets and the test dataset in each experiment 100
4.3 Misclassification rates for different consequent types (Evolve algorithm) 108
4.4 Misclassification rates for batch and incremental classification systems 109

5.1 The used variation intervals of sigma lognormal parameters and the different tested distortion combinations . . . . . 145

# Acknowledgements

# Résumé

**Apprentissage incrémental de systèmes d'inférence floue : application à la reconnaissance de gestes manuscrits**

Nous présentons dans cette thèse une nouvelle méthode pour la conception de moteurs de reconnaissance personnalisables et auto-évolutifs. La contribution majeure de cette thèse consiste à proposer une approche incrémental pour l'apprentissage de classifieurs basés sur les systèmes d'inférence floue de type Takagi-Sugeno d'ordre 1. Cette approche comprend, d'une part, une adaptation des paramètres linéaires associés aux conclusions des règles en utilisant la méthode des moindres carrés récursive, et, d'autre part, un apprentissage incrémental des prémisses de ces règles afin de modifier les fonctions d'appartenance suivant l'évolution de la densité des données dans l'espace de classification. La méthode proposée, Evolve++, résout les problèmes d'instabilité d'apprentissage incrémental de ce type de systèmes grâce à un paradigme global d'apprentissage où les prémisses et les conclusions sont apprises en synergie et non de façon indépendante. La performance de ce système a été démontrée sur des benchmarks connus, en mettant en évidence notamment sa capacité d'apprentissage à la volée de nouvelles classes. Dans le contexte applicatif de la reconnaissance de gestes manuscrits, ce système permet de s'adapter en continue aux styles d'écriture (personnalisation des symboles) et aux nouveaux besoins des utilisateurs (introduction à la volée des nouveaux symboles). Dans ce domaine, une autre contribution a été d'accélérer l'apprentissage de nouveaux symboles par la synthèse automatique de données artificielles. La technique de synthèse repose sur la théorie Sigma-lognormal qui propose un nouvel espace de représentation des tracés manuscrits basé sur un modèle neuromusculaire du mécanisme d'écriture. L'application de déformations sur le profil Sigma-lognormal permet d'obtenir des tracés manuscrits synthétiques qui sont réalistes et proches de la déformation humaine. L'utilisation de ces tracés synthétiques dans notre système accélère l'apprentissage et améliore de façon significative sa performance globale.

# Abstract

We present in this thesis a new method for the conception of evolving and customizable classification systems. The main contribution of this work is represented by proposing an incremental approach for the learning of classification models based on first-order Takagi-Sugeno (TS) fuzzy inference systems. This approach includes, on the one hand, the adaptation of linear consequences of the fuzzy rules using the recursive least-squares method, and, on the other hand, an incremental learning of the antecedent of these rules in order to modify the membership functions according to the evolution of data density in the input space. The proposed method, Evolve++, resolves the instability problems in the incremental learning of TS models thanks to a global learning paradigm in which antecedent and consequents are learned in synergy, contrary to the existing approaches where they are learned separately. The performance of our system had been demonstrated on different well-known benchmarks, with a special focus on its capacity of learning new classes. In the applicative context of handwritten gesture recognition, this system can adapt continuously with the special writing styles (personalization of existing symbols) and the new needs of the users (adding new symbols). In this specific domain, we propose a second contribution to accelerate the learning of new symbols by the automatic generation of artificial data. The generation technique is based on Sigma-lognormal theory, which proposes a new representation space of handwritten forms based on a neuromuscular modeling of writing mechanism. By applying some deformations on the Sigma-lognormal profile of a given gesture, we can obtain synthetic handwritten gestures that are realistic and close to human deformation. Integrating this data generation technique in our systems accelerates the learning process and significantly improves the overall performance.

# Chapter 1

# General Introduction

Machine learning has become during the last decades one of the main fields in computer science and applied mathematics. More and more focus is being placed on this domain thanks to the progress in the calculation capacity of the current computers. The learning approaches have been used in wide range of applications like prediction, simulation, diagnostic and classification problems. All these approaches aim at providing models that represent the essential aspects of the existing systems. Machine learning techniques become indispensable solution when the relationship between system inputs and outputs is difficult to understand and cannot be easily described using mathematical or knowledge-based models (sometimes referred as "white-box" models). The conception of machine learning models is based on a set of real data (pairs of input-output vectors) and not on physical laws of the system. Models obtained using data-driven learning algorithms are usually considered as "black-box" models.

Classification techniques represent a very active topic in machine learning. They appear frequently in many application areas, and become the basic tool for almost any pattern recognition task. Several structural and statistical approaches have been proposed to build classification systems from data. Traditionally, the classification system is trained using a learning dataset under the supervision of an expert that controls and optimizes the learning process. The system performance is fundamentally related to the learning algorithm and the used learning dataset. The learning dataset contains labeled samples from the different classes that must be recognized by the system. In almost all the learning algorithms, the learning dataset is visited several times in order to improve the classification performance which is usually measured using a separated test dataset. The expert can modify the parameters and the setting of the learning algorithm and restart the learning process until obtaining an acceptable performance. Then, the classification system is delivered to the final user to be used in real applicative contexts. The role of the classifier is to suggest a label for each unlabeled sample provided by the application. Typically, no learning algorithms are available at the user side, as it can be noted from Figure 1.1 that illustrates the two-phase life cycle of a traditional classification system.

The main weakness in the above-mentioned conception paradigm is that the knowledge base is constrained by the learning dataset available at the expert side and cannot be extended based on the data provided at the user side. The static nature of the classifier at the user side leads to two principal drawbacks:

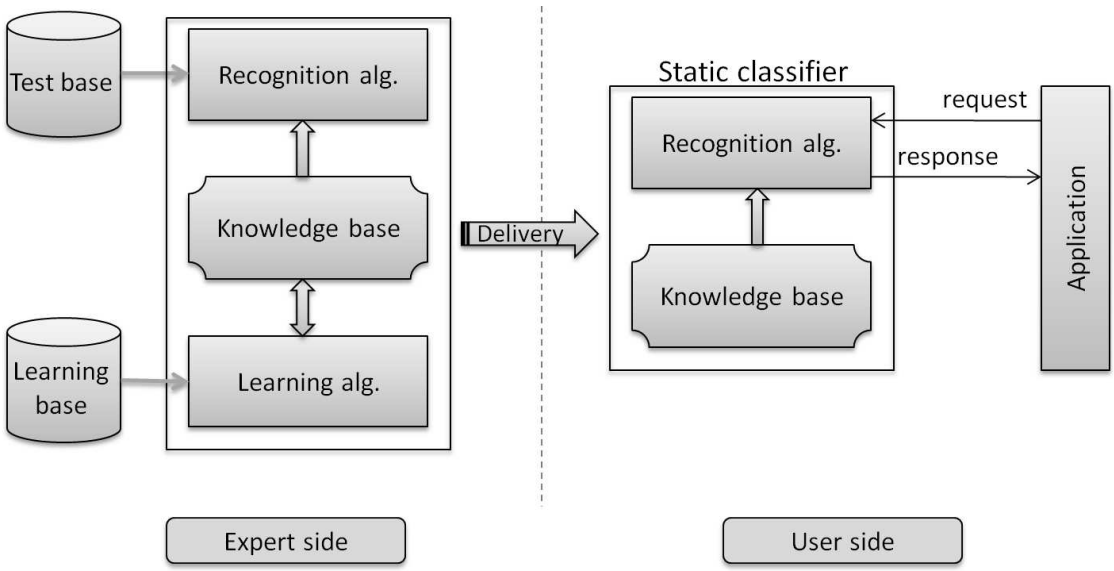



Figure 1.1: The life cycle of static classifiers with distinct learning and operation phases

- The first drawback is that the classifier cannot be adapted according to the data on the user side that represent more accurately the specific real context than the learning dataset at the expert side. It is usually very difficult to get an exhaustive learning dataset that can cover all the characteristics of the real system in all the possible contexts, especially in uncertain and imprecise environments, like in handwriting, face, or speech recognition systems. Thus, it can be very useful to enrich the classifier knowledge base by new samples of the learned classes obtained from the very specific operation environment.

- The second drawback is that the classifier cannot take into consideration new classes that were not represented in the learning dataset at the expert side. The set of classes recognizable by the classifier is defined at the expert side and cannot be modified by the application. Although the learning becomes impractical if a very large set of classes is used, it is still hard in many applicative contexts to predict all the possible classes required by the user. Moreover, the user needs may evolve and the classifier must cope with these new needs and integrate new classes into its knowledge base.

These drawbacks increase the need for new type of classification systems that can learn, adapt and evolve in lifelong continuous manner. These classification

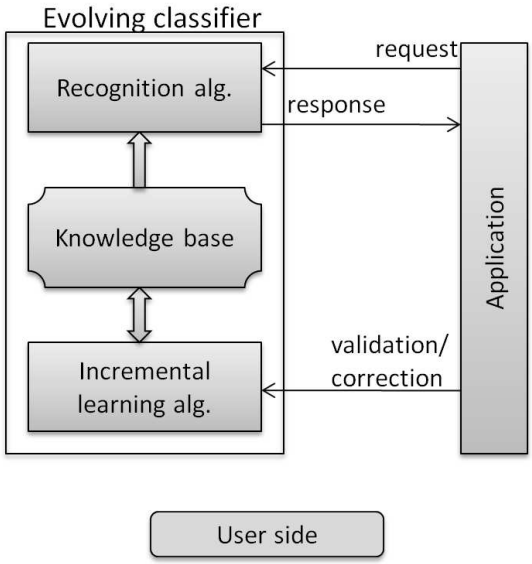



Figure 1.2: Simultaneous operation and learning (incremental) processes in evolving classification systems

systems will be referred in this thesis as "evolving classifiers". As we can see from Figure 1.2, an incremental learning algorithm is used to learn from the data samples provided by the user after sending a validation or a correction signal in order to confirm or change the label suggested by the classifier. Contrary to the traditional paradigm illustrated in Figure 1.1, there is no separation between the learning (or the conception) phase and the operation phase in evolving classification systems. One of the key features in evolving classifiers is that the incoming samples may bring in new unseen classes that are learned by the classifier without destroying its knowledge base or forgetting the existing classes.

In evolving classification systems, the learning process may, either, start from scratch at the user side so that no initial learning phase is required and the system only learns the classes required by the user, or, an initial learning phase is performed by an expert as in Figure 1.1 using a learning dataset and an initial set of classes.

In the field of handwriting recognition which is the central activity of our research group, static classification systems are the dominated approach that had been used in the handwriting-related applications during the last two decades. However, a great need for evolving classifiers appears in many of these applications due to the different handwriting styles and the very diverse and changing user requirements.

Obviously, static classifiers cannot cope with this dynamic, imprecise and open environment where the recognition system is fundamentally related to human behaviors and habits.

Therefore, we have placed a special focus in our research group activity on engaging the final user in the system conception, learning and operation process. Motivated by this context, the goal of my PhD work has consisted in doing a deep investigation in the evolving classification techniques that allow an efficient and a high interactivity with the user. Although few adaptation techniques had been proposed to personalize writer-independent handwriting recognition systems, evolving classifiers have never been used, to the best of our knowledge, in any published work related to handwriting recognition. The difference between adaptive and evolving classifiers is clarified in the next chapter.

We can take, for example, a concrete case of a handwriting related application where evolving classifiers are indispensable: a handwriting recognition system used to interpret handwritten gestures that are assigned to commands related to the operation system or a specific application. These "handwritten shortcuts" replace the classical keyboard shortcuts in touch screen computers like white-boards, tablet PCs, PDA... It is very practical to allow the user to choose his own set of gestures and to add new ones at any moments in order to assign them to new interactive commands according to the evolving of his needs. Obviously, traditional pre-learned and static classification systems cannot be used in this applicative context. The learning process must be online, incremental and lifelong and starts from scratch at the user side.

The learning of new gestures starts by few samples provided by the user. However, the learning of the existing gestures continue for each new incoming sample in order, on the one hand, to improve the knowledge of the classifier about these gestures since only few samples are used to start learning them, and, on the other hand, to adapt the classifier to any modifications in the user writing/sketching style.

The global contribution of this work consists in proposing an evolving classification system structure and an efficient learning algorithm that allows incremental, lifelong, stable and fast learning. Our evolving approach outperforms the existing ones thanks to several improvements on the structure and the learning algorithm. Our classification approach is based on a Takagi-Sugeno (TS) fuzzy inference sys-

tem. The rule-based structure of this system allows some flexibility in tuning its knowledge based, which makes it suitable for incremental learning. The classification system consists of a set of local linear regression models defined in different sub-spaces localized by the antecedent part of the rules.

The first contribution that we present in this thesis is a new enhanced antecedent structure in TS models. This enhanced structure allows more precise data covering and improves the overall classification performance.

We present a complete incremental learning algorithm for evolving TS classifiers; it contains three sub-algorithms:

- an incremental clustering algorithm to add new rules when new data clusters are detected,
- an antecedent adaptation in order to modify the rules' membership functions according to the evolution of data distribution in the input space,
- and an adaptation of linear consequences of the rules using the recursive least-squares method.

The main challenge in the learning algorithm is the stability of the consequent learning in spite of the continuous modifications on the antecedent structure. This problem becomes more crucial with the proposed enhanced and more sophisticated antecedent structure.

As a second principal contribution of this PhD work, we propose two different strategies to cope with the abovementioned stability problem. The fist one is presented in a learning algorithm that we call "Evolve". The instability effects on the consequent learning are reduced in Evolve by adopting a delayed antecedent adaptation policy using a partial memory buffer.

The second strategy consists in proposing a new learning paradigm in which the output error is used to control the antecedent learning and to avoid useless antecedent modifications. Thus, antecedents and consequences are learned in synergy in this novel global learning paradigm, contrary to the existing approaches where they are learned independently. This novel learning paradigm is first implemented for general evolving problems, so that in the new version of the learning algorithm (Evolve+) the global output error is added to the antecedent leaning formulas. Moreover, this global error is replaced by the risk of misclassification (or the

confusion degree) in the second implementation of the novel paradigm, Evolve++, which is dedicated to evolving classification problems with more than two classes.

In addition to the theoretical contributions mentioned above that can be used in any evolving classification problem, we propose a significant contribution related to our applicative context which is evolving handwritten gesture recognition systems. Based on our expertise in the handwriting domain, we enhance the incremental learning process by integrating into the system an efficient handwriting data generation technique. This domain-specific contribution aims at accelerating the learning of new handwritten symbols by the automatic generation of artificial data. The generation technique is based on Sigma-lognormal theory, which proposes a new representation space of handwritten forms based on a neuromuscular modeling of writing mechanism. By applying some deformations on the Sigma-lognormal profile of a given gesture, we can obtain synthetic handwritten gestures that are realistic and close to human deformation. Integrating this data generation technique in our systems accelerates the learning process and significantly improves the overall performance.

The rest of the manuscript is organized as follows: the **second chapter** comprises first definitions of some important terms related to incremental learning, and then a description of the main characteristics of a typical incremental learning approach and the constraints that should be respected. Next, we give an overview of the state-of-the-art works related to our problem. This overview includes several adaptive classification techniques, several incremental clustering methods, and some well-known existing evolving classification approaches. After a description of the general structure of a fuzzy inference system (FIS), a special focus is placed on the existing evolving approaches that are based on this structure. At the end of the chapter, we make a general comparison between the presented evolving FIS approaches and we introduce then the main improvement directions of our proposed approach.

We present in the **third chapter** the architecture of our TS FIS evolving system with a focus on the new antecedent structure. The different elements of our incremental learning algorithm, Evolve, are then explained in details. After that, the learning instability issue is discussed and the buffer-based solution used in Evolve is presented.

In the second part of the third chapter, we present a second strategy to cope with instability effects and to get faster and more stable learning. As mentioned above, a novel global learning paradigm is proposed in which the antecedent learning is driven by the output error. A straightforward general implementation of this paradigm is presented in Evolve+ algorithm. Furthermore, a more efficient implementation specific to classification problems is explained and its confusion-driven strategy is used in Evolve++ learning algorithm.

The different learning algorithms presented in the third chapter are experimentally evaluated in the **fourth chapter**. The tests are essentially carried out on a handwritten gesture dataset called "Sign" that has been collected as a part of this PhD work. Besides, some experiments are performed on different famous benchmarks. The performance of our evolving system using the different algorithms is compared to a well-known evolving TS model. In addition to the rapidity of incremental learning from scratch, a special focus is placed on the capacity of learning new classes.

The **fifth chapter** is dedicated to the motor-based handwriting generation technique that we propose in this thesis and its integration in the incremental learning process of an evolving handwriting recognition system. The chapter starts by a brief description of the existing handwriting generation techniques that are generally based on geometric deformations, with a special focus on previous related work performed in the IMADOC research group. The Sigma-lognormal theory is then concisely explained and two different strategies of handwriting generation based on this motor representation space are presented. The role of artificial data generation in accelerating the incremental learning of evolving handwriting classifiers is experimentally demonstrated at the end of the fifth chapter. The proposed handwriting generation method using motor-based deformations is compared to the existing IMADOC technique which is based on geometric deformations. The effectiveness of applying deformations on the motor parameters of handwriting symbols is demonstrated by the experiments results.

# Chapter 2

# Evolving Classification Systems: State Of The Art

## 2.1 Preamble

This chapter will first focus on several theoretical definitions related to the incremental learning domain. The role of these definitions is to clarify the general requirements that must be satisfied in incremental learning approaches, in addition to some particular criteria that are directly related to our application. Next, we present some well-known adaptive classification approaches and we explain the difference between these dynamic classifiers and the evolving classifier that we are looking for.

As mentioned in the precedent chapter, the classification approach in this thesis is based on a fuzzy rule-based system. The learning algorithms of these systems contain usually a clustering step in order to define the zone of influence of each rule. Obviously, this clustering step must be done in an incremental manner if the entire learning algorithm is incremental. For this particular reason, we present in Section 2.4 a brief overview of different incremental clustering methods, including the e-Clustering method that is used in our algorithm as we will see in the next chapter.

We present then in Section 2.5 several state-of-the-art evolving systems that have shown good results in different incremental learning problems. We principally focus on evolving fuzzy rule-based systems to which our model belongs. In order to offer the reader an easy understanding of these approaches and the contribution of our model compared with them, we give a brief description of the basic points of fuzzy inference systems in Section 2.5.2 before presenting the different approaches. A global comparison between these systems is then presented and the potential improvement directions are discussed.

An advanced topic related to the incremental learning theory is briefly presented at the end of this chapter, although it had not been studied in our work and stands as future perspective. This topic concernes the idea of evolving representation space and incremental feature selection or extraction methods. Two of these methods are mentioned in Section 2.6.

## 2.2 Definitions

Machine learning is a well-established discipline that had been investigated through the last four decades by many researchers. A lot of learning schemes and algorithms

had been proposed and their good performance had been proven in different real-world application areas.

Nevertheless, the learning algorithm that we are looking for in this thesis has some new characteristics that make it different from the well-know learning algorithms (like SVM, MLP, HMM, etc.). In this section, we will first define some important keywords like "incremental", "online", "evolving" ... and using these definitions we will give a clear description about the learning model that we are looking for at the end of the chapter.

### 2.2.1 Batch learning vs. Incremental learning

The difference between these two learning modes is that the learning dataset in batch mode is predefined and available since the beginning of the learning process. Thus, the dataset can be visited several times in order to optimize a specific goal function and to establish the model structure. Most of traditional algorithms use this type of learning, such as back propagation algorithm for example.

On the other hand, the whole learning dataset is not necessarily available in an incremental learning mode. The system must learn from each data sample separately, without knowing either the next learning data samples or the past ones. After observing each data sample, the system makes decision in order to change its structure and modify its parameters, based only on the presented sample and the current system state.

Let's suppose $x_i, i = 1, 2, .., n$ represent the learning data samples, $M$ refers to the learned system, and $f$ refers to a given learning algorithm. Then, the difference between batch and incremental learning can be simply defined as follows:

$$\text{Batch: } M_i = f(x_1, x_2, .., x_i), \qquad \text{Incremental: } M_i = f(M_{i-1}, x_i)$$

In practice, this learning mode is usually used when the whole learning dataset is unknown and the samples arrive separately. Nevertheless, an incremental learning mode is sometimes used to learn huge datasets where it is impractical to propagate it several times during the learning process.

In some learning approaches, the learning dataset is introduced to the model as several sequential subsets. Traditional batch algorithm can be applied on each subset of data, but the whole learning process is often considered as incremental.

We call this in-between mode "incremental batch learning", while the incremental mode is sometimes called "sample-wise" or "sample-per-sample" learning.

Our classification problem requires an incremental learning approach because the classes are not predefined and there is no fixed learning dataset. It must also be "sample-wise" in order to learn using few samples.

### 2.2.2 Offline learning vs. Online learning

In the offline learning mode, the learning and operating phases are completely separated in the model life-cycle. The model is trained on a specific dataset, in batch or incremental mode, and then implemented to operate in a real environment without any further learning activities during operation. While in the online mode the model learns from any new available data as it operates, and the data might exist only for a short time.

Using the aforementioned definitions, we can distinguish different possible combinations:

- Offline and batch learning: all training data are available throughout the learning process, and it is possible to revisit the entire dataset as times as necessary. In addition, the system will not be put into use before the end of learning phase. This represents the learning mode used in most classification problems.

- Offline and incremental learning: This learning mode is used when the size of the training set is very important and it is very expensive to keep and revisit all the data. Again, the system will not be used before the end of learning.

- Online and incremental learning: the specificity of this mode is that in addition to the incremental aspect, the system is used during learning. Our learning algorithm belongs to this category.

### 2.2.3 Active learning vs. Incremental learning

In active learning approaches, a pool of unlabeled samples is available since the beginning of the learning process. The question is to choose the most representative samples in order to get them labeled by the user. Then, based on these manually labeled samples, other samples can be labeled automatically in an intelligent way

using some heuristics, and both labeled and unlabeled samples are used in the learning process.

In some active learning methods, the learning is done in an incremental mode where the labeled samples are introduced to the system incrementally, and the criterion used to choose the next samples to be labeled is based on system performance.

On the other hand, in sample-per-sample incremental learning methods, the system does not choose the next data and the global data distribution is not a priori known as in active learning. Thus, the main question in active learning approaches is "how to choose the best sample to be labeled?", whereas the question in incremental learning is "how to deal with the received sample?". Some interesting works on active learning can be found in [17], [18] and [19].

It had been mentioned that in our application context the upcoming learning samples are completely unknown and are progressively provided by the user, which makes active learning approaches not relevant to our work.

### 2.2.4 Instance memory vs. Concept memory

As mentioned in [20], incremental learning systems can have two types of memory:

- Concept memory: a memory for storing concept descriptions and knowledge extracted from data;
- Instance memory: a memory for storing data instances.

In an online incremental learning algorithm, the training set is not available a priori because the learning examples arrive over time. Although online learning systems can continuously update and improve their models, not all of them are necessarily based on a real incremental approach. For some systems the model is completely rebuilt at each step of learning using all available data, they are systems with "full instance memory" [21]. On the other hand, if the learning algorithm modifies the model using only the last available learning example, it is called a system with "no instance memory" [22]. A third category is that of systems with "partial instance memory", which select and maintain a reduced subset of learning examples to use them in the next learning step [23].

Therefor, an incremental learning system must be based on an instance memory, a concept memory, or both of them. Thus, we obtain three different types of systems

| **Concept memory** | **Instance memory** | **Incremental learning cost** | **Classification cost** | **Memory cost** |
|---|---|---|---|---|
| Yes | No | Medium | Low | Low |
| Yes | Yes (full or partial) | High | Low | High |
| No | Yes (full or partial) | Low | High | High |

Table 2.1: Incremental learning methods categories depending on their used memories

as shown in Table 2.2.4. This table shows the characteristics of these systems in term of incremental learning cost (time required to learn new sample), classification or recall cost (time required to recognize new sample), and memory space cost.

For the second family (with concept memory and instance memory), any traditional batch learning algorithm can be used in an incremental mode by storing all past training data samples, adding them to new ones, and re-applying the algorithm.

One example of the third family is a basic incremental K-Nearest Neighbor (KNN) classifier. The incremental learning process of a new sample consists in adding this sample to the data set (low incremental learning cost); an exhaustive search is required to recognize a give sample (high classification cost); and obviously the memory space needed is relatively big (high memory cost).

In our lifelong incremental learning approach, the use of full instance memory is not practical for three reasons:

- The accumulated dataset can attain a very big size after a long period of operation of the system.
- The system is designed to be used on personal computers that may not have the required large memory space.
- The incremental learning process using full instance memory will be low and not practical after a period of operation, especially on personal computers

### 2.2.5 Incremental learning criteria

An incremental learning algorithm is defined in [24] as being one that meets the following criteria:

- it should be able to learn additional information from new data;

- it should not require access to the original data (i.e. data used to train the existing classifier);
- it should preserve previously acquired knowledge (it should not suffer from *catastrophic forgetting*, significant loss of original learned knowledge);
- and it should be able to accommodate new classes that may be introduced with new data.

Many of the existing "incremental learning" algorithms are not truly incremental because at least one of the mentioned criteria is violated. These criteria can be briefly expressed by the so called "plasticity-stability dilemma"[25], which is that a system must be able to learn in order to adapt to a changing environment but that constant change can lead to an unstable system that can learn new information only by forgetting everything it has so far learned.

### 2.2.6 Discussion

As mentioned in the introduction, the applicative context on which we are focusing in this thesis is a handwritten gesture recognition system that can evolve to include new classes (gestures). The final user must have the possibility of adding new gestures at any moment, and with few provided samples, the classifier must learn the new classes and enhance its performance using the future data.

It is obvious that the learning mode required for this need must be incremental and not batch. Even "incremental batch learning" (see Section 2.2.1) is not suitable, and the learning must be "sample-wise" in order to make the classifier evolve (new class, more accuracy) after each single sample.

Moreover, online learning (see Section 2.2.2) is required in this context because the learning is done with the final user and the classifier must be operational (with acceptable accuracy degree) during the lifelong incremental learning. However, the learning process does not have to run in real-time, a parallel thread can be used and the updates on the classifier can be committed upon the end of the current learning step.

Learning is supposed to be supervised and all data samples will be labeled in our application. One manner to do the labeling task automatically and with no explicit questions is to observe the user actions and validate the last classifier response to

the last input sample if no "undo" action is raised by the user. After this automatic validation, the sample-wise incremental learning algorithm can be applied.

While the learning here is lifelong and new classes can be added to the classifier after years of operation, a full instance memory (see Section 2.2.4) is not practical and should be avoided. The incremental learning approach proposed in this thesis is based on a full concept memory and no instance memory. Nevertheless, a short partial memory has been experimented in a modified version in order to deal with some stability issues, as we will see in the next chapter.

Thus, any incremental learning method proposed to meet the application requirements must satisfy the criteria mentioned in Section 2.2.5. However, further recommendations must also be considered. For example, problem-dependent parameters must be avoided as much as possible, since the learning is done by the user without the presence of an expert, and learning must start from scratch.

Hence, the main characteristics required in our learning algorithm can be summarized by the following points:

- to be incremental and sample-wise;
- to be online;
- not to be based on full instance memory;
- to be able to learn from scratch and with few data;
- to be able to add new classes;
- to avoid problem-depended parameters.

These points will be respected and considered in the decisions of the choices of the system and the algorithms on which our approached will be based, as we will note in the rest of this chapter.

## 2.3 Adaptive classification systems

As explained before, some classification systems can be designed to be dynamic so that it can be automatically modified during its operation phase, contrary to traditional classifiers that remain intact during its use. However, these dynamic

classifiers may either only change their parameters and tune them to get better performance, or change their structure as well in order to add new outputs (classes).

We can distinguish two main types of incremental learning algorithms: algorithms for parameter incremental learning and algorithms for structure incremental learning. The incremental learning of parameters can be considered as "adaptation" algorithm. The structure in such systems is fixed and initialized at the beginning of the learning process, and the system parameters are learned incrementally according to newly available data. For the rest of this thesis, classifier with incremental parameter learning will be called "adaptive classifiers", and those with incremental learning of parameters and structures will be called "evolving classifiers".

The set of classes that can be recognized by adaptive classification systems does not evolve, neither by adding new needed classes, nor by deleting useless ones. Nevertheless, some internal adjustment can be continuously made in order to improve the system performance using the new data samples obtained by using the system.

Although offline batch learning can be used to train these classifiers given that the classification problem is predefined, the continuous adaptation capability is still very important for two reasons:

- the adaptation data samples are related to the very end user or the specific real-world context and they can specialize the classifier and improve it,
- given the difficulty of building large learning datasets in several application areas, the adaptive classifier can be initialized by a small dataset, and be then improved thanks to data received during the operation phase.

We give in the remainder of this section some examples of adaptation approaches that have been used in several application areas, with a special focus on handwriting recognition systems.

### 2.3.1 AdaBoost-based learning

In the context of handwriting recognition systems, the authors of [26] proposed to replace the static writer-independent system by a writer-dependent one in order to get higher recognition accuracy without requiring extensive training time on the part of the user. They propose to bootstrap the writer-dependent training process with a pre-trained writer-independent recognition system. Contrary to other methods that

adapt writer-independent systems to a specific writing style, the authors of [26] do not modify the writer-independent system but incorporate it (their scores) into the writer-dependent learning process. The classification scheme is based on Pairwise AdaBoost framework that uses a weighted linear combination of weak classifiers to make a strong one. Generally, a weak classifier can be learned on a subset of training samples, a subset of classes, a subset of features or any combination of these three choices. In [26], each weak classifier is learned on a pair of classes (binary classifier) and on one single feature and all the available training samples. The classification of a given symbol depends on the class that wins the most Pairwise comparisons. The writer-independent recognizers can be then used in two ways:

- their scores for the given symbol are introduced as additional features, and a weak classifier is associated with each one;
- or as pre-recognition step in order to prune symbol pairs (so their associated weak classifiers) so that only symbols that are present in the n-best list given by the writer-independent recognize are kept. This pre-recognition step increases the accuracy of the final recognizer by providing a reduced list of candidate symbols and reduces the number of Pairwise weak classifiers.

Experiments in [26] showed that for 48 different symbols, it took user about 45 minutes to enter 10 instances per symbol, and then an writer-dependent AdaBoost classifier is learned with integrating Microsoft recognizer for this list of symbols. Recognition accuracy had been increased from 91.3% using Microsoft writer-independent recognizer to 95.8% using the writer-dependent one. It is important to mention that several epochs are required by AdaBoost in order to estimate the weak classifiers weights, which makes it unsuitable for sample-wise incremental learning.

### 2.3.2 Dynamic Time Warping

In [1], the authors propose a handwriting recognition system based on Dynamic Time Warping (DTW) distance. The training phase is done by the selection of a prototype set that represents all the characters using some clustering methods. In order to recognize a given symbol, the point-to-point correspondences between the symbol and all the prototypes are calculated and the symbol is classified according to K-nearest neighbor principal (see Figure 2.1).

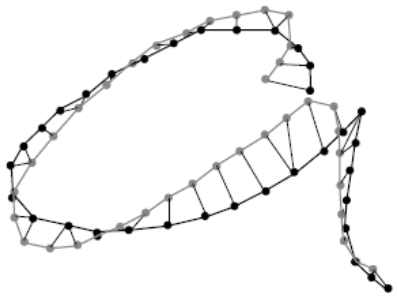


Figure 2.1: The point-to-point correspondence of two characters established with a DTW-algorithm [1]

An adaptation mechanism is adopted in this approach in order to turn the writer-independent system into a writer-dependent one to increase the recognition accuracy. The proposed adaptation strategy includes:

- adding new prototypes,
- inactivating confusing prototypes that are more harmful than useful,
- and reshaping existing prototypes.

The reshaping method is based on a modified version of Learning Vector Quantization (LVQ). If the nearest prototype belongs to the same class as the input symbol, the data points of the prototype are moved towards the corresponding points of the input symbol. Results showed that this simple mechanism can significantly reduce the misclassification rate. However, it is clear that this method is specific for the adaptation of handwriting recognition classifiers and cannot be applied to any classification problem.

### 2.3.3 Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

The authors of [2] proposed an extension of the classic neural networks error back-propagation method based on the gradient descent. In this extension, a hybrid learning method that combines the gradient method and the least squares method is used. Each epoch of this hybrid learning procedure is composed of a forward pass and a backward pass. In the forward pass, the parameters (weights) in the output layer are identified using the least squares estimation. Then, the error rates

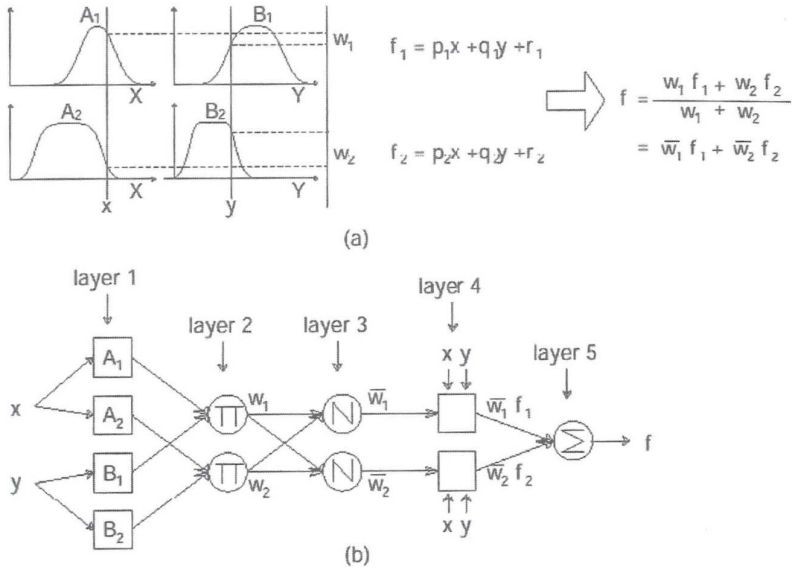



Figure 2.2: Functional equivalence between a traditional fuzzy inference system and ANFIS [2]

propagate from the output toward the input end and the parameters in the hidden layer(s) are updated by the gradient descent method.

Based on the functional equivalence between adaptive networks and fuzzy inference systems (as demonstrated in Figure 2.2), the hybrid learning method is used to learn a fuzzy inference system of Takagi Sugeno's type. Thus, consequent parameters are first estimated using least squares, and premise parameters are then updated using error-propagation gradient descent (as shown in Table 2.2).

Despite the wide use of ANFIS in different applications, we can note that it cannot be used in an evolving learning context because of its fixed structure. However, it is still a robust learning algorithm for adaptive classification systems.

| | **Forward Pass** | **Backward Pass** |
|---|---|---|
| **Premise Parameters** | Fixed | Gradient Descent |
| **Consequent Parameters** | Least Squares Estimate | Fixed |
| **Signals** | Node Outputs | Error Rates |

Table 2.2: Learning steps in ANFIS

### 2.3.4 Adaptation by Adjustment of Prototypes (ADAPT)

The ADAPT method [3] has been used as an automatic online adaptation mechanism to the handwriting style of the writer for the recognition of isolated handwritten characters. The classifier that is based on a fuzzy inference system is first trained on a multi-writer dataset, and then adapted to fit to the handwriting style of the specific writer that uses the system.

The system in ADAPT is based on a zero-order Takagi-Sugeno fuzzy inference system. It consists of a set of fuzzy rules, each fuzzy rule makes a link between an intrinsic model in the feature space and the corresponding label. Each intrinsic model is defined by a of fuzzy prototype $P_i$ in $n$ dimensions.

$$\textbf{IF } \vec{X} \text{ is } P_i \textbf{ THEN } s_1^i = a_{i1} \textbf{ and ... and } s_c^i = a_{ic} \textbf{ and ... and } s_K^i = a_{iK},$$

where $\vec{X}$ is the feature vector of the handwritten character. As each prototype can participate to the description of each class, the rule $R_i$ has numeric conclusions connecting $P_i$ with each class $c = 1..K$ by a prototype score $s_c^i$. The $a_{ic}$ is a weight that expresses the participation of $P_i$ in the description of the class $c$.

The ***AD**aptation by **A**djustment of **P**roto**T**ypes* (ADAPT) method allows to modify all the prototypes of the FIS by re-centering and re-shaping them for each new example that is inputted. This is done according to their participation in the recognition process. Conclusions adaptation is done using back-propagation algorithm over the examples stored in a specific buffer (i.e. Incremental batch learning). Thus, the two adaptation processes that are applied on the premise of an FIS are:

- **Prototype re-centering:** the principle is inspired by the Learning Vector Quantization (LVQ) algorithm. In ADAPT, all prototype's centers are updated for each new example, and the update of each center $\mu_i$ of the prototype $P_i$ must improve the score of each class. In this way, there are three reasons to have a significant displacement $\Delta\vec{\mu}_i$: the class score $s_c$ is different from the one expected; the participation $s_c^i$ of the prototype to the final decision is high; and the activation of the premise is high. Equations (2.1, 2.2) gives the prototype updating with the proposed method:

$$\vec{\mu_i} \Leftarrow \vec{\mu_i} + \lambda * \delta' * (\vec{X} - \vec{\mu_r}) \tag{2.1}$$

$$\delta' = \beta_i * \left( \sum_{c=1}^{C} (b_c - s_c) * s_c^i \right), \tag{2.2}$$

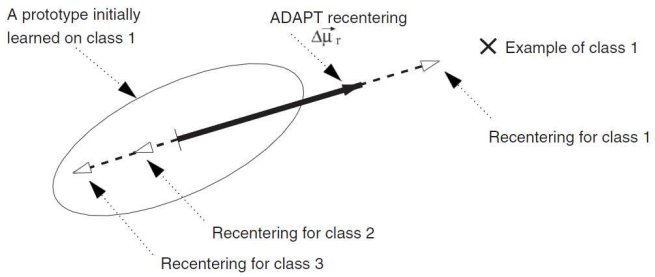



Figure 2.3: Principle of the ADAPT compromises between the optimizations of all classes [3]

with $b_c$ the expected class score for $s_c$ : 1 if $c$ is the example class and 0 otherwise.

The adaptation parameter $\lambda$ lies between 0 and 1. It controls the amplitude of the displacement and thus the adaptation rate.

- **Prototype re-shaping:** The re-centering of the prototypes allows to fit the new localization of the data in the input space. To better represent the repartition of these new data, the shape of the prototypes must also be adapted. ADAPT proposes an iterative formula to estimate the inverse covariance matrix that is used to calculate the activation degree of the prototype by a given data sample (so it represents the zone of influence of the prototype):

$$Q_i^{-1} \Leftarrow \frac{Q_i^{-1}}{1-\alpha\delta'} - \frac{\alpha\delta'}{1-\alpha\delta'} \cdot \frac{(Q_i^{-1}\vec{m}) \cdot (Q_i^{-1}\vec{m})^T}{1+\alpha\delta'(\vec{m}^T Q_i^{-1}\vec{m})}, \tag{2.3}$$

with $\vec{m} = \vec{X} - \vec{\mu_r}$ and $\alpha$ is the learning rate.

### 2.3.5 Discussion

The aforementioned techniques were useful in their contexts, but the incremental learning approach that we are looking for must be able to learn new classes in an incremental lifelong learning mode. Nevertheless, when a new available example from an existing class is introduced to the system, an adaptation method can be adopted. In our incremental learning approach presented in this thesis, the adaptation of the premise (or antecedent) of the used fuzzy inference system architecture is inspired by ADAPT method.

## 2.4 Incremental clustering

Here we will focus on incremental clustering because our classifier is based on fuzzy rule-based system, in which the rule creation is usually considered as a clustering problem as we will see later.

The clustering process aims at extracting some statistical knowledge from a set of data and eventually dividing it into subsets (clusters) so that data points that belongs to the same cluster share similar characteristics in some sense. The result of this process is essentially finding cluster centers, and the zone of influence of each cluster. The membership value of a given data point to a cluster can be either crisp (1 or 0) or fuzzy (between 0 and 1). Clusters are allowed to overlap in many cases, especially when fuzzy membership values are used.

In the traditional clustering methods, where the process is done in an offline batch mode, the entire dataset is visited several times in order to find the optimal clustering, either by optimizing an objective function or by using some clustering validity criteria. Examples of these methods are K-means clustering, hierarchical clustering, fuzzy C-means clustering, Gath-Geva clustering, etc [27].

In an incremental clustering mode, a decision must be taken for each new data point. So each new point may either reinforce an existing cluster, and eventually changes its characteristics (i.e. its center and zone of influence), or trigger the creation of a new cluster. The main difference between incremental clustering methods is the criterion that is used to make the decision between these two choices. In the rest of this section, we will give some incremental clustering methods that belong to different families according to this criterion.

### 2.4.1 Distance-based Incremental clustering

Most of published incremental clustering methods can be considered as distance-based. In these methods, a threshold value is directly or indirectly defined and used to decide whether a new cluster must be created or not depending on the minimum distance between the new data point and the existing cluster centers. Some examples of these methods are detailed below.

#### 2.4.1.1 Adaptive Resonance Theory (ART) Networks

The ART paradigm [28] can be described as a type of incremental clustering. It is a self-organizing network that had been designed in respect to the stability-plasticity dilemma, so that the weights change according to new knowledge without forgetting the old one. Moreover, the structure of ART evolves in order to add new categories (classes) when they are needed. As explained above, the mechanism of ART is based on measuring the matching (the distance) between the input vector and the category vectors (cluster centers). A typical ART network (see Figure 2.4) consists of:

- A comparison layer (or field): that receives the input signal and transfers it to the recognition layer;
- A recognition layer: each neuron in this layer passes back to the comparison layer a negative signal proportional to the category match with the input signal;
- A reset module: that in its turn compares the strength of the recognition match to a specific threshold value called "the vigilance parameter". If this threshold is met, the winner category vector is modified according to the input signal (adaptation). Otherwise, the next neuron in the recognition layer is considered. Thus, the search procedure continues until the vigilance parameter is satisfied by a recognition match. Finally, if no match can meet the threshold, a new neuron in the recognition layer is assigned and adjusted to match the input signal creating a new category. In other words, the reset module works like a novelty detector.

#### 2.4.1.2 Vector Quantization

The concept of vector quantization had been proposed in signal processing field in order to as a data compression technique. However, it can be used as a clustering method as it finds a set of "centroids" that optimally represents all the data points. Vector quantization algorithm is equivalent to k-means method implemented in an incremental mode. It starts by selecting the first $C$ data points as cluster centers (it is important to mention that the number of clusters is defined a-priori in the algorithm). Then, for each new data point, the distances between this point and all

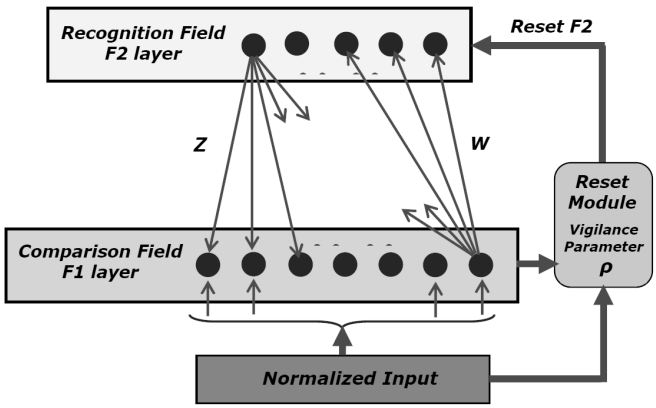



Figure 2.4: Basic ART structure [4]

the centers are calculated and the closest center is selected as the winner neuron. Thus, the winner neuron is updated using the following formula:

$$\vec{v}_{win} = \vec{v}_{win} + \eta \, (\vec{x} - \vec{v}_{win})$$

where $\eta$ is a learning parameter, and $\vec{x}$ is the new data point.

Some extension of VQ use a modified version of this formula with a specific term that considers the neighborhood between the winner neuron and the other neurons, like SOM [29] and Neural Gas [30]. The ability of creating new clusters in ART can be combined to the adaptation ability of these methods in order to obtain a hybrid incremental clustering techniques like VQ-ART [31], SOM-ART, etc.

#### 2.4.1.3 Evolving Clustering Method (ECM)

ECM [5] is defined as a fast, one-pass, maximum distance-based clustering method. ECM is used for dynamic estimation of the number of clusters in a set of data, and to find their centers in the input data space. ECM depends strongly on a specific value *Dthr* that defines the maximum allowed distance between an example point in a cluster and the cluster center. Obviously, the clustering parameter *Dthr* affects the number of clusters to be estimated. Existing clusters are updated with new examples through changing their centers' positions and increasing their cluster radius. ECM algorithm in 2D space is illustrated in Figure 2.4.1.3.

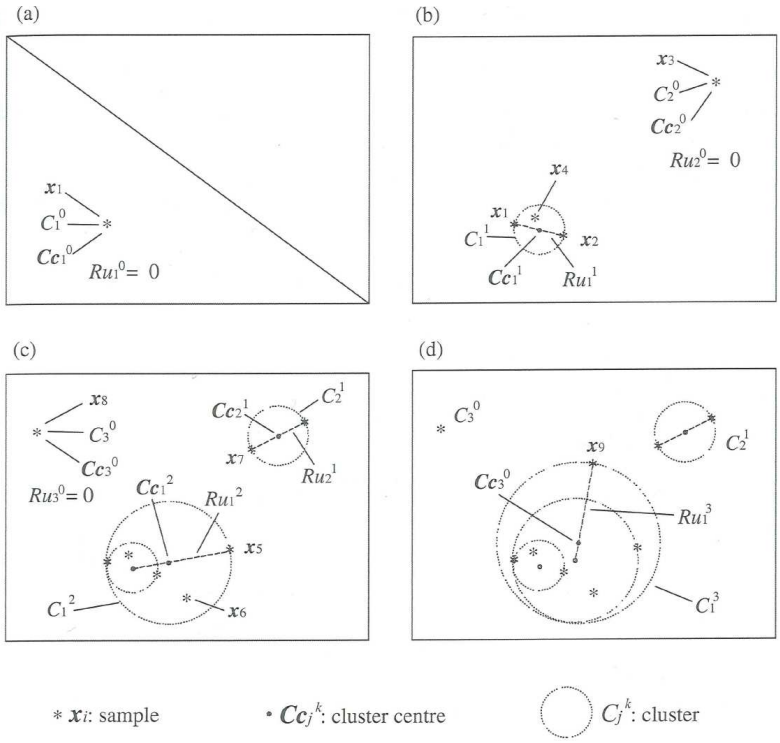



Figure 2.5: ECM process with consecutive examples $x_1$ to $x_9$ [5]

## 2.4.2 Density-based Incremental clustering

### 2.4.2.1 Graph-based Incremental clustering methods

Some clustering methods are based on a graph representation of data points where each node is associated to a sample in the feature space, while to each edge is associated the distance between nodes connected under a suitably defined neighborhood relationship. A cluster is thus defined to be a connected sub-graph, obtained according to particular criteria for each specific algorithm [32]. The advantage of graph-based clustering methods is the capability of modeling the spatio-temporal relationship between data points in a more precise manner comparing to statistical and distributions estimations.

One of the most known graph-based incremental clustering approaches is the RepStream algorithm [6]. RepSteram is one-pass incremental clustering algorithm that uses two sparse graphs of k-nearest neighbor vertices (data points). The first graph, also called sparse graph, contains the last data points introduced to the algorithm and represents the closeness between these points. The second graph, the representative sparse graph, contains representative vertices that have been chosen

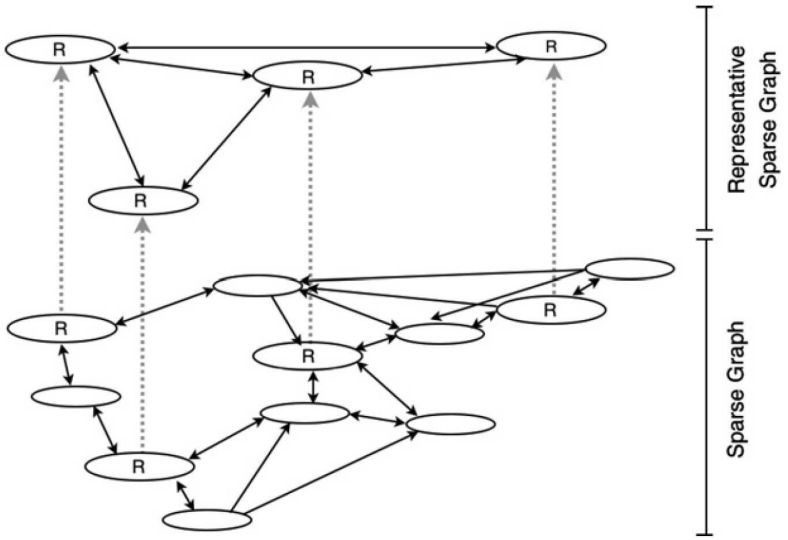



Figure 2.6: The relationship between the sparse graph and the representative sparse graph [6]

so far from the sparse graph using density-based criteria. These two graphs and the relationship between them are illustrated in Figure 2.6. Representative sparse graph is used to make decisions at higher level in an efficient way without requiring access to the all the data points. The representative vertices can be merged and they give the shape of clusters. More details about Repstream can be found in [6]. Similar algorithms of density-graph-based incremental clusters have been proposed, like MOSAIC [33] and Chameleon [34].

#### 2.4.2.2 Recursive Density Estimation (eClustering)

In general, when introducing a new training sample in an online learning mode, it will either reinforce the information contained in the previous data and represented by the current clustering, or bring enough information to form a new cluster or modify an existing one. The importance of a given sample in the clustering process can be evaluated by its *potential* value. The potential of a sample is defined as inverse of the sum of distances between a data sample and all the other data samples [35]:

$$Pot(\vec{x}(t)) = \frac{1}{1 + \sum_{i=1}^{t-1} \|x(t) - x(i)\|^2} \tag{2.4}$$

A recursive method for the calculation of the potential of a new sample was introduced in [36] under the name of eClustering method. The recursive formula

avoids memorizing the whole previous data but keeps - using few variables - the density distribution in the feature space based on the previous data:

$$Pot(\vec{x}(t)) = \frac{t-1}{(t-1)\alpha(t) + \gamma(t) - 2\zeta(t) + t - 1} \tag{2.5}$$

where

$$\alpha(t) = \sum_{j=1}^{n} x_j^2(t) \tag{2.6}$$

$$\gamma(t) = \gamma(t-1) + \alpha(t-1), \quad \gamma(1) = 0 \tag{2.7}$$

$$\zeta(t) = \sum_{j=1}^{n} x_j(t)\eta_j(t), \quad \eta_j(t) = \eta_j(t-1) + x_j(t-1), \quad \eta_j(1) = 0 \tag{2.8}$$

Introducing a new sample affects the potential values of the centers of the existing clusters, which can be recursively updated by the following equation:

$$Pot(\mu_i) = \frac{(t-1)Pot(\mu_i)}{t - 2 + Pot(\mu_i) + Pot(\mu_i)\sum_{j=1}^{n} \|\mu_i - x(t-1)\|_j^2} \tag{2.9}$$

If the potential of the new sample is higher than the potential of the existing centers then this sample will be a center of a new cluster.

### 2.4.3 Discussion

The common drawback in three aforementioned distance-based incremental clustering methods is the strong dependence on the minimum inter-clusters threshold value. All these method require a precise setting of this value which is not always possible given that the learning is performed at the user side in one-pass mode and trial-and-error methods cannot be used to optimize this threshold value. It is difficult to have a problem-independent optimal minimum inter-clusters distance since the dimensions of the data space can be very different depending on the given problem. A bad setting of this threshold may lead to either over-clustering, a data cluster is divided into several small ones, or under-clustering, different clusters are erroneously merged to form one big cluster. Another major disadvantage of distance-based incremental clustering is the sensibility to noise and outlier points. Many false clusters can be created because of these points that do not belong a real new cluster but sufficiently distant from the existing ones so that they wrongly trigger the cluster creating process.

Therefore, we believe that density-based techniques are much more suitable for incremental clustering. Contrary to distance-based one, density-based do not dependent on an absolute threshold distance to create new cluster. They rely on density measures to make a global judgment on the relative data distribution.

Incremental learning problems can be divided into two categories: stationary and non-stationary. In non-stationary problems, the environment is considered very dynamic and the data points progressively lose their significance so that the learning process should principally focus on newly incoming points and ignoring old ones. In stationary problems, old and new points are equitably considered in the learning process. Therefore, only the spatial information of the data points is relevant to the incremental learning process and the temporal information can generally be ignored.

While the applicative principal motivation of this work, evolving handwriting recognition systems, can be more or less seen as stationary problem, the spatio-temporal representation used in graph-based incremental clustering techniques is not preferred. Moreover, the growing graph of data points can lead to serious memory issues in a lifelong learning process.

For all the above-mentioned reasons, the eClustering density-based approach represents the most favorable choice to be used as a part of our evolving system. We can sum up the advantages of this technique as by the following three points:

- it does not require a user setting of a problem-dependent threshold value,
- as a density-based approach, it is very less sensitive to noise and outliers,
- and all the density measures are recursively estimated using few variables and without requiring access to precedent data points.

More details about eClustring method and its role in our evolving TS fuzzy inference system are presented in the next chapter.

## 2.5 Overview of some evolving systems

After wide investigation and analysis of the evolving classification systems proposed during the last decade, we found out that most of them can be classified in two categories:

- Ensemble Systems

- Fuzzy Rule-based Systems

The evolving process in the former, where the system is represented by a synergy of several classifiers using different combination mechanisms, consists in the addition of new classifiers for new incoming data. While the manageable structure of the latter allows the learning algorithm to add new rules to the system in order to add new classes.

We give in this section a brief description about some approaches from both categories, with more focus on the second one to which the system proposed in this thesis belongs.

### 2.5.1 Evolving Ensemble Systems

The goal of ensemble system is to generate a divers set of classifiers so that each of which realizes different decision boundaries. The class decision of the ensemble for a given instance can be taken using different majority voting strategies. Using ensemble systems in incremental learning mode can be straightforward by training new classifier(s) for each incoming dataset. However, special efforts are usually done in order to guarantee a minimum level of diversity between the classifiers of the ensemble, either in the manner of selecting the instances to be learned by each classifier, or by explicitly considering this diversity in the learning of new classifiers. Two representative examples of incremental ensemble learning are presented below.

#### 2.5.1.1 Learn++

Learn++ [24] by inspired on AdaBoost method where instead of creating new clusters for unseen data, it creates a set of "weak" classifiers. The prediction capacity of Learn++ is based on the synergistic performance of the ensemble of classifiers obtained using a specific voting strategy. Each new available learning dataset is sampled into several learning subsets according to a specific distribution of probability, and each learning subset is used independently to train a weak classifier. The sampling process in Learn++ is done in a sequential manner, and the distribution of probability is built in a way that instances that are difficult to classify receive higher weights to increase their chance of being selected into the next learning subset. As other ensemble of classifiers methods, Learn++ requires each weak learner to generate only a rough estimate of the decision boundary. Learn++ is implemented in

[24] using MLP as the base classifier.

#### 2.5.1.2 Growing Negative Correlation Learning

Negative Correlation Learning (NCL) [37] of neural network-based ensemble classifier aimes at producing diverse neural networks in an ensemble by inserting a penalty term into the error function of each individual neural network in the ensemble. This term is called "negative correlation penalty term", and its role is to make the ensemble as diverse as possible so that the difference among its neural networks make them adapt in different ways to new data. NCL can be used in incremental learning in two ways:

- **Fixed NCL**: In this method, for any new data subset, the whole ensemble is retrained with this subset using the weights obtained from the previous training as initial weights. The advantage of this method is that some neural networks can adapt fast and better than others, but the problem is that the neural networks may suffer from catastrophic forgetting.
- **Growing NCL**: Similar to Learn++, for each new incoming data subset, a new neural network is trained and inserted in the ensemble. Only the new neural network is trained with the new data subset. The old NNs do not receive any new training on the new data, but their outputs for the new data are calculated in order to learn the new NN using negative correlation learning method.

We can note that negative correlation is indirectly used in Learn++ when it creates new classifier using a data subset that is sampled according to a probability of distribution related to the scores of the other classifiers. However, the construction of a new classifier to new data does not have interaction with the classifiers created to previous data as in Growing NCL. However, these methods of incremental ensemble learning that generate new classifiers for new data suffer from an essential problem when introducing new unseen classes that is the "outvoting" problem. The problem is due to old classifiers voting for instances of new classes on which there were not trained, and this may result in incorrect and unstable final scores. This specific drawback had been discussed in [38].

It is important to mention that even though these methods satisfy the main criteria incremental learning by not requiring access to old data and not suffering from

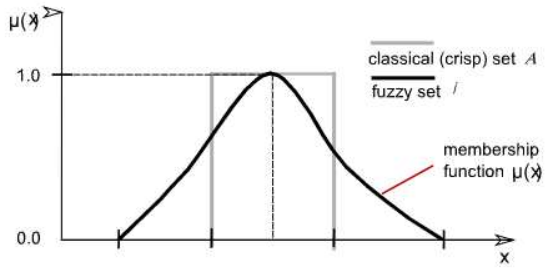



Figure 2.7: The difference between crisp and fuzzy sets

catastrophic forgetting, they can be only used in incremental batch learning where data is introduced in small sequential batches and not in sample-wise incremental learning mode.

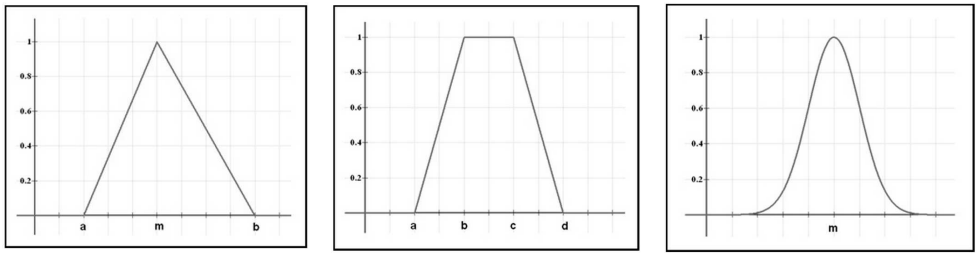
### 2.5.2 Fuzzy inference systems

In the rest of this section, we give a detailed description of fuzzy inference systems, before presenting some well-known evolving FIS and analyzing their drawbacks.

#### 2.5.2.1 Fuzzy set theory

The difference between fuzzy sets and classic sets is that values can belong to a fuzzy set with various degrees, whereas they either belong or not to a classical set. Based on this definition, each fuzzy set is represented by a membership function that is used to determine the membership degree (between 0 and 1) of any given value to that set (see Figure 2.7). Different mathematical functions are used to represent the membership function. The characteristics of the membership function might significantly change the behavior of the fuzzy systems. The most used membership functions are (see Figure 2.8) :

1. Triangular function: which is defined by three values (a,b,m);
2. Trapezoidal function: this membership function is defined by four values (a,b,c,d);
3. Gaussian function: is the most used type and defined by a center value $m$ and a spread value $k$.




(a) (b) (c)

Figure 2.8: Different membership functions: (a) Triangular, (b) Trapezoidal, and (c) Gaussian.

Other possible membership functions are cited in [39]. In addition to the definition of a fuzzy set, the other important part of the fuzzy theory proposed by Zadeh is the fuzzy operations between fuzzy sets, such as union, intersection, complement operations. More details about this point can be found in [40].

#### 2.5.2.2 Fuzzy inference systems structures

The first Fuzzy Inference System (FIS) (or fuzzy rule-based system) has been proposed by Zadeh [41]. It is composed of a set of rules written as in the following form:

IF (antecedent) THEN (consequence)

The difference between the fuzzy IF-THEN rules and the "crisp" rules used in traditional expert systems is that the antecedent part and eventually the consequence part in the former are represented by membership degrees to some fuzzy sets and not by exact values as in the latter. Thanks to this important feature the input signal in FISs can activate several rules with different degrees instead of one rule with full activation as in the classic rules. The parameters of the membership functions in the antecedent or the consequence can be estimated using some automatic data-driven mechanisms, which makes FISs able to achieve high precision degrees while preserving an acceptable transparency level due to the open rule-based structure. For this reason, FISs are usually categorized as gray-box models.

The structure and the mechanism of FISs can be illustrated by the toy example in Figure 2.9 (from Matlab's tutorial). We can divide the fuzzy inference mechanism of an FIS into three steps:

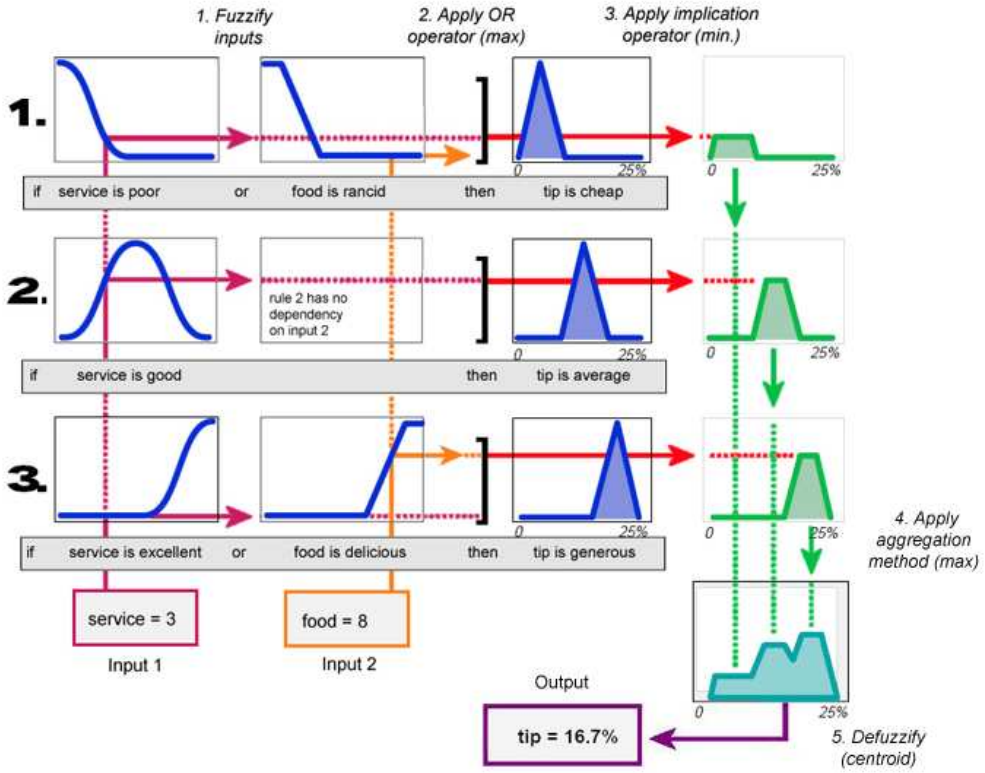



Figure 2.9: The different steps of the fuzzy inference mechanism for a given toy example

1. Fuzzify crisp inputs to get the fuzzy inputs.
2. Apply the fuzzy rules to get fuzzy outputs (this can be divided into two sub-steps: implication and aggregation).
3. Defuzzify the fuzzy outputs to get crisp outputs.

Two basic groups of FISs had been used during the last decades. These two groups differ by the type of the consequent part.

- **Mamdani models**

  This type is similar to the original FIS proposed by Zadeh and presented before, but Mamdani [42] has introduced some enhancement on it. A fuzzy rule in Mamdani FIS is written as follows:

$$IF\ (x_1\ is\ A_1)\ AND\ (x_2\ is\ A_2)\ ...\ Then\ (y\ is\ B)$$

where $x_1, x_2, ...$ are the input variables, $A_1, A_2, ...$ are the input fuzzy sets, and $B$ is the output fuzzy set.

The Mamdani model represents a mapping between input fuzzy sets and output fuzzy sets. The main advantage of this model is the high transparency and interpretability capabilities. However, Mamdani models generally achieves less accuracy compared to Takagi-Sugeno models. The learning of Mamdani model consists in finding the best fuzzy partitioning or fuzzy clustering of the input and the output space.

- **Takagi-Sugeno models**

  The fuzzy rules in Takagi-Sugeno (TS) models are defined as follows:

$$IF \ \ (x_1 \ \ is \ \ A_1) \ \ AND \ \ (x_2 \ \ is \ \ A_2) \ \ ... \ \ Then \ \ y = l(x_1, x_2..) \tag{2.10}$$

  where $l(x_1, x_2..)$ is the consequent function which represents the main difference between Mamdani and TS models. The fuzzy output sets in the consequent part of the former are replaced by a polynomial function $l$ of the input variables in the latter. Different type of functions (linear, non-linear, singleton ... etc) can be used, this point will be discussed later in this chapter.

  Generally, the rule activation degree (the aggregation of the membership degrees) is added as an additional term in the consequence function of that rule in order to estimated the output crisp value. It is worth mentioning that no defuzzification step is needed in the fuzzy inference mechanism of TS models.

A hybrid Mamdani-Takagi-Sugeno model had been presented in a recent study [43], in the motivation of having a new model that meets the precision and the interpretability criteria.

#### 2.5.2.3 Takagi-Sugeno FIS

As explained above, a Takagi-Sugeno FIS is defined by a set of fuzzy rules in which the antecedent part represents a fuzzy partitioning or clustering of the input space, and the output is calculated using a regression polynomial model over the input vector weighted by the antecedent activation degree. However, different architectures of TS models have been adopted in previous works, so that each of which can be

used for a specific problem and has different characteristics. Some of these variants are briefly presented in this section.

**2.5.2.3.1 Single vs. multiple output** Since TS models have been used in wide variety of applications, the type of system output was adapted to the class of application. Generally, Multi-Input-Single-Output (MISO) TS models are used for regression, prediction or approximation problems. In this architecture, a single real output value is considered and can represent the function value for the given input vector, or the system state at the instant $t$, where the input vector represent the systems states (values) at previous instances ($t-1$, $t-2$, ... , $t-k$). The fuzzy rules of this architecture are similar to Equation 2.10 .

Nevertheless, MISO TS models are not suitable for classification problems especially problems with more than two classes, which is the case of the target application of this thesis. For classification problems, TS models with Multi-Input-Multi-Output (MIMO) architecture are generally used. The fuzzy rules of this architecture are defines as follows:

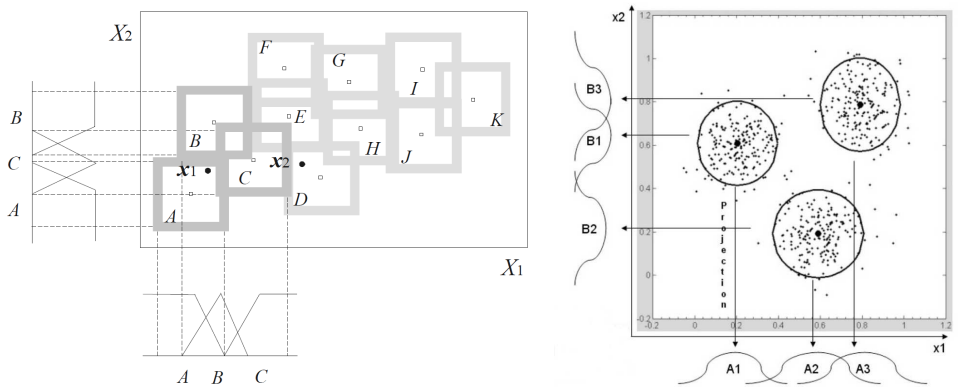
$$IF\ (x_1\ is\ A_1)\ AND\ (x_2\ is\ A_2)\ ...\ THEN\ y_1 = l_1(x_1, x_2..),\ ...,\ y_k = l_k(x_1, x_2..) \tag{2.11}$$

where $k$ represents the number of classes (outputs).

For the learning of MIMO TS models, learning pairs $(\vec{x}, \vec{\bar{y}})$ are introduced to the learning algorithm, where $\vec{x}$ is the input vector, and $\vec{\bar{y}}$ represents the real label vector so that it contains 1 for the real class of $\vec{x}$, and 0 for the rest.

**2.5.2.3.2 Antecedent structure** Let $\vec{x} = \{x_1, x_2, ..., x_n\}$ be the input vector of a TS model that consists of $n$ features, each of which describes a specific aspect of the observed samples. The antecedent part of a TS models' rule consists then of an aggregation of at most $n$ fuzzy sets or membership functions. The membership function $A_i$ is defined on the axis corresponding to the input variable or the feature $x_i$.

As one can notice from Equation 2.11, the input membership functions of the antecedent part are aggregated using the logical AND operation in order to calculate the overall activation (or firing) degree of the fuzzy rule. The AND operation can be implemented using a conjunction operation between the fuzzy sets. Several t-norm




(a) (b)

Figure 2.10: (a) Hyper-rectangle zones of influence obtained by the aggregation of triangular membership functions [5], (b) hyper-spherical (or elliptical) zones of influence obtained by the aggregation of Gaussian membership functions [7].

functions are usually used to implement the AND operation in TS models, such as minimum t-norm or product t-norm.

Different types of membership functions can be used in TS models as shown in Section 2.5.2.1. The conjunction of the membership functions that are defined on the axes (features) of the input space results in a hyper fuzzy zone of influence associated to the fuzzy rule. The form of this fuzzy zone is related to the used membership function of the antecedent part. Figure 2.10 illustrates two different types of the fuzzy zone: the hyper-rectangular zones used in DENFIS system (Section 2.5.3.4), and hyper-spherical ones used in FLEXFIS system (Section 2.5.3.1).

If we consider the case of Gaussian functions, we can rewrite the fuzzy rules of TS models so that the antecedent part can be represented by a fuzzy zone of influence with hyper-spherical shape. This zone of influence can be characterized by a center and a radius value. In the rest of this thesis, we will use the word "prototype" to refer to the fuzzy zone of influence of a fuzzy rule.

In data-driven design of TS models, the antecedents of the fuzzy rules are formed using batch or incremental fuzzy clustering methods over a learning data set. These clustering methods aim at finding the prototypes' centers and estimating the radius value in order to optimally cover the input data cloud(s).

We can express the firing degree of the antecedent part via a specific distance that represents the closeness degree between the input vector and the fuzzy prototype (equation 2.12).

$$\textbf{Rule}_i : \textbf{IF } \vec{x} \textit{ is close to } P_i \textbf{ THEN } y_i^1 = l_i^1(\vec{x}), ..., \ y_i^k = l_i^k(\vec{x}) \tag{2.12}$$

where $P_i$ represents the fuzzy prototype associated to the rule $i$.

In the existing systems, the zone of influence has often a hyper-spherical form. It has then the same radius for all the feature space dimensions. More sophisticated form allows the zone radius to have different values for different dimensions, which results in hyper-elliptical zones so that the ellipses' main axes are parallel to the feature space axes.

For hyper-spherical or axes-parallel hyper-elliptical prototypes, the firing degree can be computed depending on the prototype center $\vec{\mu_i}$ and the radius value $\sigma_i$ (the same value in all the dimensions for the former, and different values for the latter).The value of the firing degree $\beta_i(\vec{x})$ of the rule $i$ can then be computed as follows:

$$\beta_i(\vec{x}) = \prod_{j=1}^{n} \exp(-\frac{\|x - \mu_i\|_j^2}{2\sigma_{ij}^2}) \tag{2.13}$$

This prototype structure will be the adopted as the basis of the antecedent structure in our model as we will see later.

**2.5.2.3.3 Inference process** When using TS models in a classification problem, the inference process, applied to get the class of a given input vector $\vec{x}$, consists of three steps:

- The activation (or firing) degree of each rule $\beta_i(\vec{x})$ in the model is calculated (using equation 2.13, for example). It must then be normalized as follows:

$$\bar{\beta}_i(\vec{x}) = \frac{\beta_i(\vec{x})}{\sum_{j=1}^{r} \beta_j(\vec{x})} \tag{2.14}$$

  where $r$ represents the number of rules in the model.

- the sum-product inference is used to compute the system output for each class:

$$y^m(\vec{x}) = \sum_{i=1}^{r} \bar{\beta}_i(\vec{x}) y_i^m \tag{2.15}$$

  where $y_i^m = l_i^m(\vec{x})$ in the consequence part of the rule $i$ related to the class $m$.

- The winning class label is given by finding the maximal output and taking the corresponding class label as response:

$$class(\vec{x}) = y = argmax\ y^m(\vec{x}) \qquad m = 1, .., k \tag{2.16}$$

**2.5.2.3.4 Consequence variants** The polynomial functions $l$ in the consequent part of TS models can have different forms according to the requirements of the application, that may sometimes favor light models with less parameters, more interpretability, and thus less accuracy.

We can distinguish three different consequence structures that can be used in TS models (ordered from the less to the more sophisticated):

- Binary consequences

  The consequence function here is reduced to a binary value, so that $y_i^m$ equals 1 if the prototype $Pi$ represents the class $m$, and 0 otherwise. This very simple structure makes the TS model functionally equivalent to a mixture of Gaussians. Each fuzzy prototype is crisply associated to a specific class, and the same class can be represented by several prototypes. In order to get the class label of a given input vector $\vec{x}$, either a max-product inference is used so that the prototype with the highest firing degree is considered winner and its corresponding class is chosen as answer, or the sum-product inference (equation 2.15) is used so that the score of each class is simply the sum of the firing degrees of its prototypes.

- Singleton consequences

  TS models with this type of consequences are usually called "zero-order TS models". The polynomial consequence functions here are defined by constant values $y_i^m = s_i^m \in [0, 1]$. These values represent the participation degree of the prototype $P_i$ in the recognition of the class $m$. In this manner, each input vector $\vec{x}$ activates all the fuzzy prototypes with different degrees, and each prototype participates in the overall score of each class score with a specific degree (sometimes called "weight"). Data-driven learning algorithms can be used to adjust these weights as it will be mentioned later.

  It is notable that zero-order TS models are functionally equivalent to the well-know Radial Basis Function Networks (RBFN) [44] [45].The structures of these two models can be compared as illustrated in Figure 2.11, so that :

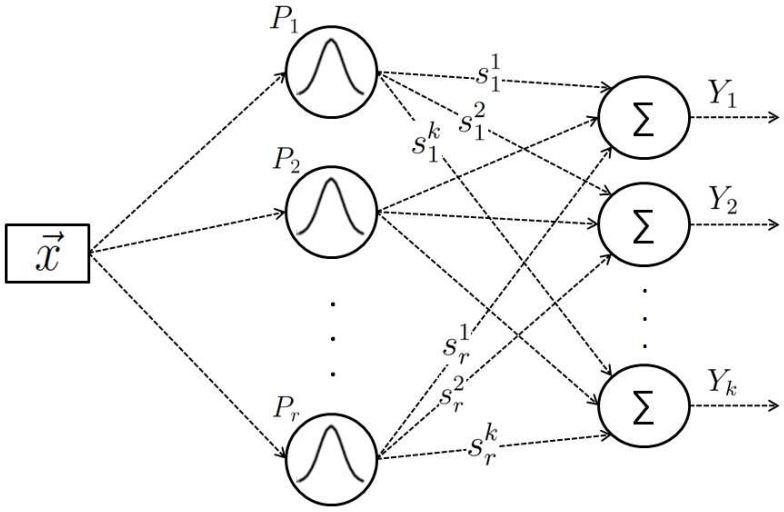



Figure 2.11: Zero-order TS model presented in the form of RBF network

- The fuzzy prototypes in TS models are equivalent to the hidden neurons in RBFN, which are multidimensional Gaussian radial basis functions.
- The singleton consequences in TS models are equivalent to the weights in RBFN between the hidden and the output layer.

- Linear consequences

  More sophisticated forms of consequences can be used in TS models in order to achieve higher precision. We will focus on the case where these consequence functions are represented by first-degree polynomial functions of the input vector. Model with such consequences structure are called "First-order TS models". Thus, the linear consequence function is written as follows:

$$l_i^m(\vec{x}) = \vec{\pi}_i^m \vec{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \tag{2.17}$$

  where $l_i^m(\vec{x})$ is the linear consequent function of the rule $i$ for the class $m$.

  For the purpose of comparison between first-order and zero-order TS, Figure 2.12 shows a first-order TS model in the form of neural network, similar to the representation in Figure 2.11.

  The learning of consequences in first-order TS models consists in the estimation of the coefficients of the linear functions. The learning process aims at

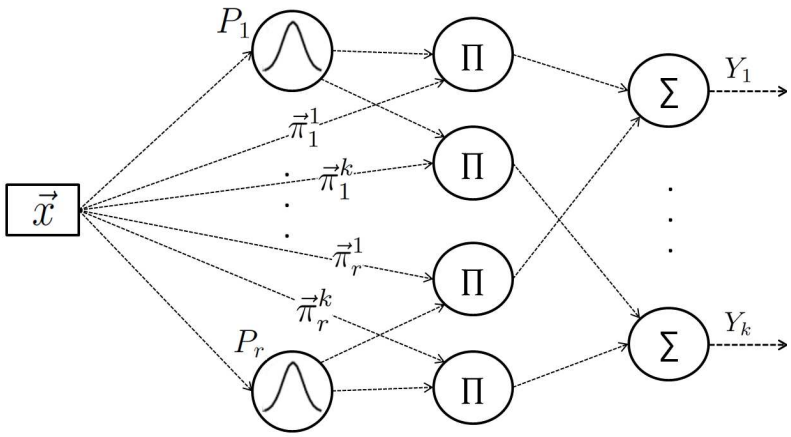



Figure 2.12: First-order TS model presented as a neural network

minimizing the error between the estimated values $y_k$ and the real values $\bar{y}_k$ for each learning pair $(\vec{x}, \vec{\bar{y}})$. This minimization problem can be represented by the following error function:

$$E = \sum_{c=1}^{m} \parallel \bar{y}_c - \sum_{i=1}^{r} \bar{\beta}_i(\vec{x})\ l_i^m(\vec{x}) \parallel \tag{2.18}$$

We can note that each output is calculated as the aggregation of local sub-models represented by hyper-planes. Based on the error term in equation 2.18, we note that these hyper-planes make a regression on the function $f : \vec{x} \mapsto \vec{\bar{y}}$.

Equation 2.18 shows that input samples that barley activates the antecedent part of a specific rule, will be barley considered in the estimation of the parameters of the consequence of this rule. This local nature of the linear sub-model makes its learning possible and efficient. However, this local region in the hyper-space, defined by the activation value, may contain data points from more than one class. Thus, the advantage of the consequence sub-models is to approximate the function $f$ for the points of different classes in this local region.

Moreover, multiplying the consequence function with the activation value $\bar{\beta}_i(\vec{x})$ participates in the "linearization" of the $(\vec{x}, \vec{\bar{y}})$ pairs in the hyper-space, based on the fact that inputs with the same label will get similar activations.

Our model will be based on first-order consequent functions in order to achieve

high precision levels. However, zero-order TS will be also considered in the experimental study.

### 2.5.3 Evolving Fuzzy inference systems

After a brief explanation the principals of FISs and their different structure types, we preset some approaches that use FISs in evolving environments and incremental learning mode. The batch learning of FISs is beyond the interest of this thesis. A global comparison is drawn between the presented approaches at the end of this section and the outlines of our proposed approach are introduced.

#### 2.5.3.1 FLEXible Fuzzy Inference System (FLEXFIS)

FLEXFIS [7] [46] is based on a first-order TS system. The core idea is to extend the vector quantization method and to use it in an incremental manner to learn the antecedent part of an evolving fuzzy inference system. The proposed incremental clustering method can be considered as distance-based clustering and inspired by ART networks, but it is enhanced by three techniques:

- the incorporation of the sphere of influence of clusters and proposing a distance measurement that estimates the distance of a data point to the surface of the clusters;
- the incremental update of the winner cluster center by moving it toward the input data point;
- the removal of satellites clusters and online split-and-merge strategy based on incremental cluster validation index estimation in order to enhance and simplify the premise structure.

The hyper-spheres that represent the zones of influence in FLEXFIS are created using an incremental variance estimation formula in each dimension, which leads to multidimensional ellipsoid clusters in main positions (parallel to the main axes). We can note (and it is confirmed in [47]) that FLEXFIS suffers from the problem of any distance-based incremental clustering method, which is the dependence on a vigilance parameter. This parameter is one of the key points of the whole algorithm, as it steers the tradeoff between updating already existing clusters and creating new

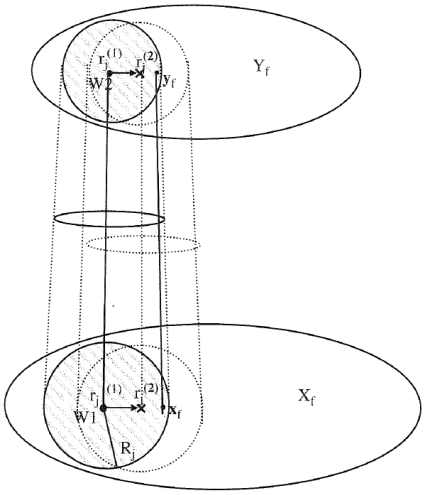



Figure 2.13: A rule in EFuNN represents an association of two hyper-spheres from the fuzzy input space and the fuzzy output space.

ones. As mentioned before, an inappropriate setting of this parameter may lead to over- or under-partitioning of data clusters. In FLEXFIS, the author proposes a heuristic estimation of this parameter according to the number of data space dimensions $P$ (considering normalized data inputs) as follows:

$$\rho = 0.3\frac{\sqrt{P}}{\sqrt{2}}$$

This formula was found using trial-and-error tuning phases with several datasets with no theoretical proof. However, the author proposes some corrections techniques like deletion and split-and-merge and apply it continuously in order to keep a correct premise structure in spite of a potential inappropriate setting of the vigilance parameter.

This learning strategy for the antecedent (premise) part is combined with the recursive least squares estimation of consequent parameters to form a complete incremental learning algorithm of an evolving classification system.

#### 2.5.3.2 Evolving Fuzzy Neural Networks (EFuNN)

EFuNN [48] is a fuzzy rule-based classification system in which each rule represents an association between a hypersphere from the fuzzy input space and a hypersphere from the fuzzy output space (see Figure 2.13) [48].

The pair of fuzzy input-output data vectors $(x_f, y_f)$ is allocated to a rule $r_i$ if $x_f$ falls in the input hypersphere and $y_f$ falls in the output hypersphere. The former condition is directly verified using a local normalized fuzzy distance, whereas the latter is indirectly verified by calculating the global output error. EFuNN algorithm starts by evaluating the local normalized fuzzy distance between the input vector and the existing rules in order to calculate the activations of the rule layer. If all the activations are below a specific sensitivity threshold, a new rule is created. Otherwise, the activation of the fuzzy output layer is calculated based on the activations of input layer and the centers of hyperspheres in output layer. The error between the system output and the real output $y_f$ is calculated and if it is higher than a specific threshold, a new rule is created. Otherwise, the centers of input hyperspheres are adjusted depending on the distance between the input vector and rule nodes, and centers of output hyperspheres are also adapted using a gradient descent as follows:

$$W1(r_j^{t+1}) = W1(r_j^t) + l(x_f - W1(r_j^t))$$
$$W2(r_j^{t+1}) = W2(r_j^t) + l(y_f - A2)A1(r_j^t)$$

The above adjustment formulas define the standard EFuNN that has the first part updated in an unsupervised mode, and the second part is a supervised mode. However, a supervised adaptation of the input hyperspheres centers is also possible based on a one-step gradient descent method as follows:

$$W1(r_j^{t+1}) = W1(r_j^t) + l(x_f - W1(r_j^t))(y_f - A2)A1(r_j^t)W2(r_j^t)$$

We can note that although EFuNN structure has been presented differently, it is functionally equivalent to Mamdani fuzzy inference system. The distance-based incremental clustering method used in EFuNN in both input and output space depends strongly on the sensitivity threshold.

#### 2.5.3.3 Generalized Fuzzy MIN-MAX (GFMM)

GFMM [8] had been presented as a kind of neural network (Figure 2.14) in which the mapping between inputs nodes and class nodes is done using hyperboxes defined in the feature space. Each hyperbox (or hyper-rectangle) is represented by two vectors $V_i$ and $W_i$ that hold the min-max points of the hyperbox. A hyperbox defines then a region of the n-dimensional pattern space, and all patterns contained within the hyperbox have full membership value. Nevertheless, for patterns located outside

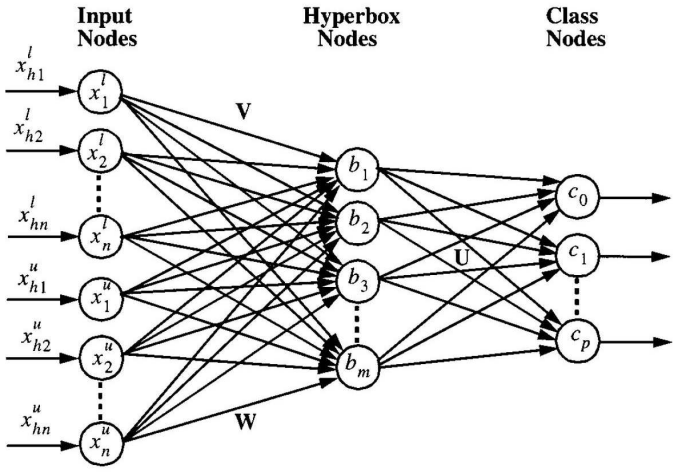



Figure 2.14: The three-layer neural network that implements the GFMM algorithm [8]

the hyperbox, a specific fuzzy membership function is used to decide whether the presented input pattern belongs to a particular class, thus whether the hyperbox is to be expanded. Incremental learning of GFMM consists of three steps: hyperbox expansion, hyperbox overlap test, and hyperbox contraction if overlap exists. An important user-defined parameter $\Theta$ is used in GFMM in order to impose a bound on the maximum size of a hyperbox. Overlaps between hyperboxes of the same class are allowed, but not between hyperboxes from different classes. The connections between the second and the third layer $U$ are represented by binary values: $u_{jk} = 1$ if the hyperbox $j$ belongs to the class $c_k$, 0 otherwise.

The main drawback in GFMM is that the prototypes are build according to a covering strategy and not a density strategy, which make it very sensible to outliers. The second weakness is that the simple connections between the hyperboxes and the class labels do not allow a fuzzy participation of each hyperbox to the recognition of each class. Similar incremental learning systems based on hyperbox prototypes had been proposed in [49] and [50].

#### 2.5.3.4 Dynamic Evolving Neural-Fuzzy Inference System (DENFIS)

The DENFIS system [5] uses Takagi-Sugeno type fuzzy inference engine where all fuzzy membership functions are triangular functions. The antecedent part in DENFIS is learned in online manner using the aforementioned Evolving Clustering Method (ECM), which is a distance-based clustering method with a predefined

cluster size parameter as explained in Section 2.4.1.3. The membership function in DENFIS is given by the following equation:

$$\mu(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases}$$

where $b$ is the value of the cluster on the $x$ dimension, $a = b - d * Dthr$ and $c = b + d * Dthr$, $d \in [1.2, 2]$, and $Dthr$ is the predefined cluster size parameter. The activation of the rule $R_i$ is calculated as follows:

$$\beta_i(\vec{x}) = \prod_{j=1}^{q} \mu_{R_{ij}}(x_j)$$

Linear consequent functions in DENFIS are learned using Recursive Least Squares (RLS) method.

#### 2.5.3.5 Evolving Takagi-Sugeno (eTS)

The structure of eTS [36] is similar to the fuzzy inference system structure presented in Section 2.5.2.2. It is defined by fuzzy rules, and each rule has two parts:

$$Rule_i : IF \ (x_1 \ is \ X_{i1}) \ AND ... AND \ (x_n \ is \ X_{in}) \ THEN \ (y = a_{i0} + a_{i1}x_1 + ... + a_{in}x_n)$$

- Antecedent part, defined by fuzzy sets connected Logical operator AND. A membership function is defined for each set in order to determine the degree of membership of any value to the set. Gaussian membership function, characterized by a center and a spread value, is usually used in eTS.
- Consequent part of linear form (singleton consequents are sometimes used).

The main idea in eTS is that the rule-base evolves to track changes in the data stream. The evolving process in eTS can be done via several actions:

- Adding new rules
- Adjusting the antecedent part of existing rules (modified centers and spreads)
- Updating consequent linear parameters.

New rule creation in eTS is based on the recursive density estimation method (eClustering) presented in Section 2.4.2.2. When a given data point is judge to have a relatively important density degree, a new rule is added to the system using this point as center with a predefined initial spread value. Adjustments in antecedent part of existing rules can be done using basic statistical formulas usually used for recursive calculation of mean and standard deviation in an incremental single-pass manner. Finally, consequent linear parameters are updated using Recursive Least Squares (RLS) method, as for DENFIS systems.

ETS model had been extended to meet with classification problems. The new approach is called "eClass". The performance of this evolving classifier had been experimentally proved in [9] to be superior to other well-known incremental systems. eClass will be used as a reference model in our experiments presented in the fourth chapter.

It is worth mentioning that our evolving approach presented in the next chapter can be seen as a significant extension of eTS (or eClass) system.

### 2.5.4 Discussion

By revising the recent relevant publications concerning evolving systems, we can note that evolving FISs can generally achieve very good results. Contrary to evolving ensemble systems, evolving FISs can be learned in sample-wise incremental learning mode, thanks to its rule-based structure. This is an important feature in the evolving handwriting classifier that we are looking for, as explained in Section 2.2.

However, it is worth mentioning that most of the evolving systems proposed during the last years had been experimented on prediction or regression problems. There was no significant focus on the performance of these systems in classification problems. Therefore, one of the contributions of this work is to study the performance of these systems in a classification problem with more than two classes, with a special attention on the behavior of the system when adding new classes.

The transparent nature of evolving FISs had been proven in different works, allowing the integration of some useful options to the systems, like rejection and confidence measures.

We sum up the main features of the evolving FISs presented in this section in Table 2.3.

| | **FLEXFIS** | **DENFIS** | **EFuNN** | **GFMM** | **eTS** |
|---|---|---|---|---|---|
| **FIS structure** | TS | TS | Mamdani | TS simplified | TS |
| **Consequent type** | 1-order | 1-order | - | 0-order (binary) | 1-order or singleton |
| **Consequent learning** | RLS | RLS | clustering | - | RLS |
| **Membership function** | Univariate Gaussian | Triangular | Univariate Gaussian | Triangular | Univariate Gaussia |
| **Antecedent learning** | Distance-based clustering | Distance-based clustering | Distance-based clustering | Distance-based clustering | Density-based clustering |
| **Prototype adaptation** | Yes | No | Yes | Yes | Yes |
| **Prototype shape** | hyper-spherical | hyper-boxes | hyper-elliptical | hyper-boxes | hyper-elliptical |

Table 2.3: The main characteristics of existing evolving FISs

Takagi-Sugeno model will be preferred in our work to Mamdani model for the next motivations:

- The high output accuracies that can be achieved using TS models compared with Mamdani models, which has been proved through different studies [51], [52].
- The fuzzy output sets in Mamdani models have no meaning in our handwriting recognition classification problem. Each handwritten symbol belongs in fact to one class, contrary to other problems where each input data can belong to different classes with different degrees. Thus, the interpretability nature of the consequent part using Mamdani models does not represent a priority in our problem.

As mentioned in Section 2.4, the drawback of distance-based clustering methods is the predefined thresholds that controls the clustering. An inconvenient setting of these thresholds results in an under- or an over-clustering. It is important in any general incremental learning algorithm to avoid problem-dependent parameters, because trial-and-error techniques cannot be used in order to find the optimal values of these parameters. For this reason, eClustering density-based incremental clustering method will be used in the incremental learning algorithm presented in this thesis in order to control the new rules (new prototypes) creation process.

The shape of the prototypes in the antecedent of FISs plays an important role in the overall performance, even when using a high degree consequent function. Generally, Gaussian membership functions fit better most of real data distributions. In some of the aforementioned evolving FISs, a fixed spread value is used to define the zone of influence of each prototype, and this same value is used for all the dimensions which results in hyper-spherical prototypes. Other approaches, like eTS and FLEXFIS, use different spread values and yet hyper-elliptical prototypes. Moreover, these values are continuously adjusted according to new incoming data. However, the activation of the antecedent part in these systems is calculated as the aggregation of one-dimensional (univariate) Gaussian distribution like in equation 2.13, and the correlations or the covariance between the different dimensions (features) are not represented in this modeling.

In this thesis, we go a step ahead in the structure of the fuzzy prototypes in TS models by allowing them to have a hyper-elliptical form non-parallel to the feature

space axes. This form enables the model to take into consideration the correlations that can exist between the features, and is represented by a covariance matrix associated to each prototype. The membership functions in this enhanced structure are represented by multi-dimensional (multivariate) probability distribution functions, as we will see in the next chapter.

Prototype adaptation is done in the existing systems either by moving the prototypes centers as in EFuNN, or by the re-estimation of the spread values as in FLEXFIS and eTS. In either distance-based or density-based incremental clustering, a new prototype is created around the incoming sample that meets the creation criterion (sufficiently far from existing prototypes, or relatively dense region). The initial position of the prototype center may not be optimal and must be updated according to future data samples. Thus, the continuous displacements of the prototype center along with the continuous updates of variance/covariance values are essential for maintaining the antecedent structure in any evolving FIS.

Therefore, the improvement in the antecedent structure of TS FIS systems presented in this thesis is coupled with recursive formulas that allow the re-estimation of the covariance matrices of the prototypes in addition to the centers' displacements formulas. All these formulas have no problem-dependent parameters. The fact of using density-based incremental clustering method to create new prototypes and parameter-free prototype adaptation formulas makes the application of our method on any new evolving classification problem is straightforward, with no need for a parameter tuning phase.

It can be noted by studying the existing evolving FISs is that the incremental learning of the antecedent part and the consequent part are completely independent. The recursive least squares (RLS) method is used in most of them for estimating the coefficients of the linear consequent functions. RLS is an optimization method that is designed to tune the parameters of linear systems under the assumption that input and output vectors characterize a specific model.

As mentioned in Section 2.5.2.3, and will be detailed in the next chapter, the input of RLS in our TS FIS is the input vector $\vec{x}$ weighted by the vector of antecedent activations. Therefore, the ideal solution to keep the exactitude of RLS learning is to use a fixed antecedent structure. Any modification performed on the antecedents during consequent learning by RLS will perturb the learning and may cause major instability problems.

Thus, the fundamental challenge in our work is to solve this contradiction between, on the one hand, adopting an improved and sophisticated antecedent structure in order to boost the overall system performance, and, on the other hand, avoiding the instability in consequent learning using RLS method.

Our learning algorithm presented in the next chapter will propose two different solutions for this problem; the first one is based on a partial memory buffer associated to the incremental learning process, whereas the second is based on minimizing the modifications on the antecedent structure by integrating the output error in the antecedent adaptation process as we will see in the next chapter.

## 2.6 Incremental Feature selection

In order to cover the different aspects related to incremental leaning and evolving classification systems, we give a brief insight into the concept of evolving data representation space. We present below the two main existing incremental feature selection and extraction approached. These approaches can be integrated to any incremental learning algorithm of an evolving classification system in order to obtain more efficient systems. It is important to mention that incremental feature selection had not been one of the centers of interest of this PhD work; it stands as important future perspective and very interesting research area.

### 2.6.1 Incremental Principal component analysis

Principal Component Analysis (PCA) is a manner of expressing the data in such a way as to highlight their similarities and differences. It transforms the original data space into a new space such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on, under the constraint that it must be orthogonal to (uncorrelated with) the preceding component. The new space dimensions can be found by calculating the eigenvectors of the data covariance matrix. It is possible to discard less significant components (eigenvectors) with low eigenvalues. Thus, the new data space will have fewer dimensions.

In the Incremental PCA [53], the feature space is updated in two manners:

- The rotation of eigenaxes;

- Dimensional augmentation by adding new eigenaxes.

Assume that an eigenspace $(\bar{x}, U, A, N)$ model is constructed based on the covariance matrix of $N$ training samples, where $\bar{x}$ is a mean vector, $U$ is a matrix whose columns represent the eigenvectors, and $A$ is a matrix whose diagonal elements correspond to eigenvalues. In an incremental learning context, the addition of a new training sample $x_{new}$ will change the mean vector and the covariance matrix and the eigenvectors have to be recalculated. For each new example, the mean vector is first updated as follows:

$$\bar{x} = \frac{1}{N+1}(N\bar{x} + x_{new})$$

Then, a residue vector $h$ is calculated:

$$h = (x_{new} - \bar{x})[1 - UU^T]$$

When the norm of the residue vector is larger than a threshold, the number of dimensions is increased by expanding the eigenspace in the direction of $h$. However, the existing eigenaxes are updated (rotated) after introducing a new example. The update formula can be found in [53].

### 2.6.2 Incremental linear discriminant analysis

Linear Discriminant Analysis (LDA) is related to principal component analysis (PCA) in that both look for linear combinations of variables which best explain the data. Nevertheless, LDA looks for components that model the difference between the classes of data and make an efficient discrimination. Whereas, PCA does not take into account any difference in classes between data and looks for the best representation. Thus, PCA can be considered as unsupervised feature selection and extraction method, and LDA is a supervised one. The core idea of LDA is to find a transformation $U$ over the input data in order to maximize the ratio of the within-class scatter matrix over the between-class scatter matrix. The transformation matrix $U$ can be calculated by an eigenvalue decomposition of $S_w^{-1}Sb$, where

$$S_w = \sum_{c=1}^{M} \sum_{x \in \{x_c\}} (x - \bar{x}_c)(x - \bar{x}_c)^T$$

is the within-class scatter matrix, and

$$S_b = \sum_{c=1}^{M} n_c(\bar{x}_c - \bar{x})(\bar{x}_c - \bar{x})^T$$

is the between-class scatter matrix, $n_c$ is the number of samples of class $c$, $\bar{x}$ is the overall mean vector of $X$, and $\bar{x}_c$ is the mean vector in class $c$.

In the incremental version of LDA, $S_w$, $S_b$, $\bar{x}$ and $\bar{x}_c$ must be updated for each new training sample $x_{new}$ with class label $k$. If $x$ is a new class sample, the within-class matrix $S_w$ does not change. Otherwise, it must be updated as follows:

$$S_w = S_w + \frac{n_k}{n_k+1}(x_{new} - \bar{x}_k)(x_{new} - \bar{x}_k)^T$$

On the other hand, the between-class scatter matrix $S_b$ can be easily re-estimated after updating the overall mean vector $\bar{x}$, and the mean vector of class $k$ as follows:

$$\bar{x} = \frac{N\bar{x}+x_{new}}{N+1}, \qquad \bar{x}_k = \frac{n_k\bar{x}_k+x_{new}}{n_c+1}$$

Where $N$ represents the overall sample number. More details about Incremental LDA can be found in [54].

## 2.7 Conclusion

After some theatrical definitions related to incremental learning field, we have presented in this chapter a quick overview of the different existing approaches that deal with this kind of problems. Several adaptive classification systems and different incremental clustering methods have been visited. Then, some well-established evolving classification systems have been cited and briefly explained, with a particular focus on rule-based approaches. We present in the next chapter our proposed evolving classification architecture that contained an important enhancement compared to the systems presented in this chapter. Moreover, different versions of our incremental learning algorithms will be presented to cope with the learning instability issues and to offer a fast and efficient incremental learning.

# Chapter 3

# Evolve(+)(+): incremental learning of evolving Takagi-Sugeno classifiers with enhanced antecedent structure

## 3.1 Introduction

In the precedent chapter, we defined the main characteristics that must be satisfied by the incremental learning algorithms of the evolving classification system that we are looking for. As we have seen, this learning algorithm must be recursive, with no or limited access to old data, and able to add new classes without suffering from the "catastrophic forgetting" problem. After a brief review of the existing well-known evolving classification approaches, we found out that Takagi-Sugeno fuzzy inference system is one of the classification architecture that proved a high flexibility and ability for incremental learning, while achieving acceptable precision levels.

We have seen as well that several evolving classifiers based on Takagi-Sugeno fuzzy inference system (TS FIS) have been recently proposed. They differ either by the structure of the antecedent or the consequences part of the system, or by the learning algorithm that are used to incrementally learn these two parts.

In this work, we propose an evolving classification system based on a TS fuzzy inference system. In order to get a high classification performance, we use a consequent structure represented by first-order linear functions (already used in some existing models like eTS, FlexFIS ...). The main originality in the structure of our model compared to the existing ones is the fact of using an enhanced antecedent structure so that the each prototype is represented by a center and a covariance matrix as we will see in Section 3.2. The covariance matrices are estimated in sample-wise incremental mode. The purpose of using this structure is to get a high-performing classifier thanks to the more precise modeling of data by the proposed antecedent structure. We describe in Section 3.2 the multi-output architecture of our fuzzy system, and then we present the enhanced antecedent structure.

The learning of TS FISs is generally divided into two independent learning processes: antecedent learning and consequent learning. For evolving TS FISs, these two learning algorithms must be incremental as explained in the precedent chapter. In our learning algorithm, the antecedent incremental learning is based, on the one hand, on a well-known incremental clustering technique to create new rules whenever it is necessary, and, on the other hand, on an adaptation technique to progressively give the prototypes their optimal position (represented by the centers), and their forms (represented by the covariance matrices).

The consequent parameters in TS FISs are traditionally learned using the recur-

sive least squares (RLS) method. This single-pass method is suitable for incremental learning mode.

As mentioned in the precedent chapter, RLS method requires a fixed antecedent structure in order to converge toward the optimal solution after each incoming sample. The more the antecedent structure is modified during the incremental learning process, the more the consequent learning will be perturbed and get instable.

Thus, the main challenge in the incremental learning of evolving TS FISs is to find a compromise between the antecedent structure and learning on one hand, and consequent structure and learning on the other hand. For this reason, many approaches adopt simple antecedent structures that may offer stable consequent learning, but in the same time they do not offer an exact modeling of the distribution of points in the data space, which reduces the overall system precision.

We will see in this chapter the difficulties of this challenge. In a beta version of our learning algorithm, in spite of the using of an enhanced antecedent structure, the overall performance cannot outperform other models with simpler antecedent structure (like eTS) because of the instability consequent learning problem explained above.

In order to overcome this problem, we present a first incremental learning algorithm that we call "Evolve", based on a memory buffer that contains the more recent incoming data. Using this buffer, the antecedent learning can be performed in incremental batch mode (see Section 2.2), so that the antecedent structure keeps unchanged for a specific period of time which allows a stable consequent learning. When the memory buffer gets full, the antecedent learning process is applied on the stored data samples. The different components of Evolve and the complete algorithm are presented in detail in Section 3.3.

Looking for a better solution to address the contradiction between having a sophisticated and efficient antecedent structure on the one hand, and allowing a stable consequent learning using RLS method on the other hand, we have developed a novel incremental learning paradigm for evolving TS FISs. Our purpose was to find a manner for minimizing as much as possible the modifications carried out on the antecedent during the incremental learning process, without sacrificing the possibility of having an efficient antecedent structure that can improve the overall performance. The core idea of this paradigm is to learn the antecedent and the consequent part in correlated manner so that the output error is used to supervise

the antecedent learning process. The goal of this supervision is to reduce as much as possible the antecedent modification for samples with low output error, and to focus instead on samples with high output error. The importance of this solution comes from the fact that in addition to reducing antecedent modifications, it accelerates the learning of "difficult" classes and confused samples so that it reduces the misclassification errors. The incremental learning algorithms "Evolve+" and "Evolve++" presented in Section 3.4 represent two different implementations of the proposed learning paradigm.

It is important to mention that, to the best of our knowledge, this is the first homogeneous incremental learning algorithm for evolving TS FISs in which the antecedent is learned in synergy with the consequences by integrating the output error in the antecedent learning. As mentioned before, the antecedent and the consequences are learned independently in all the other similar models (eTS, FlexFIS, DENFIS ...).

## 3.2 System architecture

Our system is based on first-order Takagi-Sugeno (TS) fuzzy inference system. It consists of a set of fuzzy rules of the following form:

$$\textbf{Rule}_i : \textbf{IF } \vec{x} \textit{ is close to } P_i \textbf{ THEN } y_i^1 = l_i^1(\vec{x}), ..., \ y_i^k = l_i^k(\vec{x}) \tag{3.1}$$

where $\vec{x} = \{x_1, x_2, ..., x_n\}$ is the input vector, $k$ is the number of classes(outputs), and $l_i^m(\vec{x})$ is the linear consequence function of the rule $i$ for the class $m$:

$$l_i^m(\vec{x}) = \vec{\pi}_i^m \vec{x} = a_{i0}^m + a_{i1}^m x_1 + a_{i2}^m x_2 + ... + a_{in}^m x_n \tag{3.2}$$

Singleton consequences ($l_i^m(\vec{x}) = a_i^m$) are sometimes used instead of linear functions in order to get simpler models that are called zero-order TS models. The prototype $P_i$ is defined by a center and a fuzzy zone of influence. A membership function must be defined in order to calculate the "closeness" degree between $\vec{x}$ and P (considering its center and its fuzzy zone of influence). Thus, in order to find the class of $\vec{x}$, its membership degree $\beta_i(\vec{x})$ to each fuzzy prototype is first computed. After normalizing these membership degrees, the sum-product inference is used to compute the system output for each class:

$$y^m(\vec{x}) = \sum_{i=1}^{r} \bar{\beta}_i(\vec{x}) \ l_i^m(\vec{x}) \tag{3.3}$$

where $r$ is the number of fuzzy rules in the system. The winning class label is given by finding the maximal output and taking the corresponding class label as response:

$$class(\vec{x}) = y = argmax\ y^m(\vec{x}) \qquad m = 1, .., k \tag{3.4}$$

The membership degree can be calculated in many ways. For hyper-spherical or axes-parallel hyper-elliptical prototypes, the membership degree can be computed depending on the prototype center $\vec{\mu_i}$ and the radius value $\sigma_i$ (the same value in all the dimensions for the former, and different values for the later). In this case, the Gaussian membership function is generally used. The value of $\beta_i(\vec{x})$ can then be computed as follows:

$$\beta_i(\vec{x}) = \prod_{j=1}^{n} \exp(-\frac{\|x - \mu_i\|_j^2}{2\sigma_{ij}^2}) \tag{3.5}$$

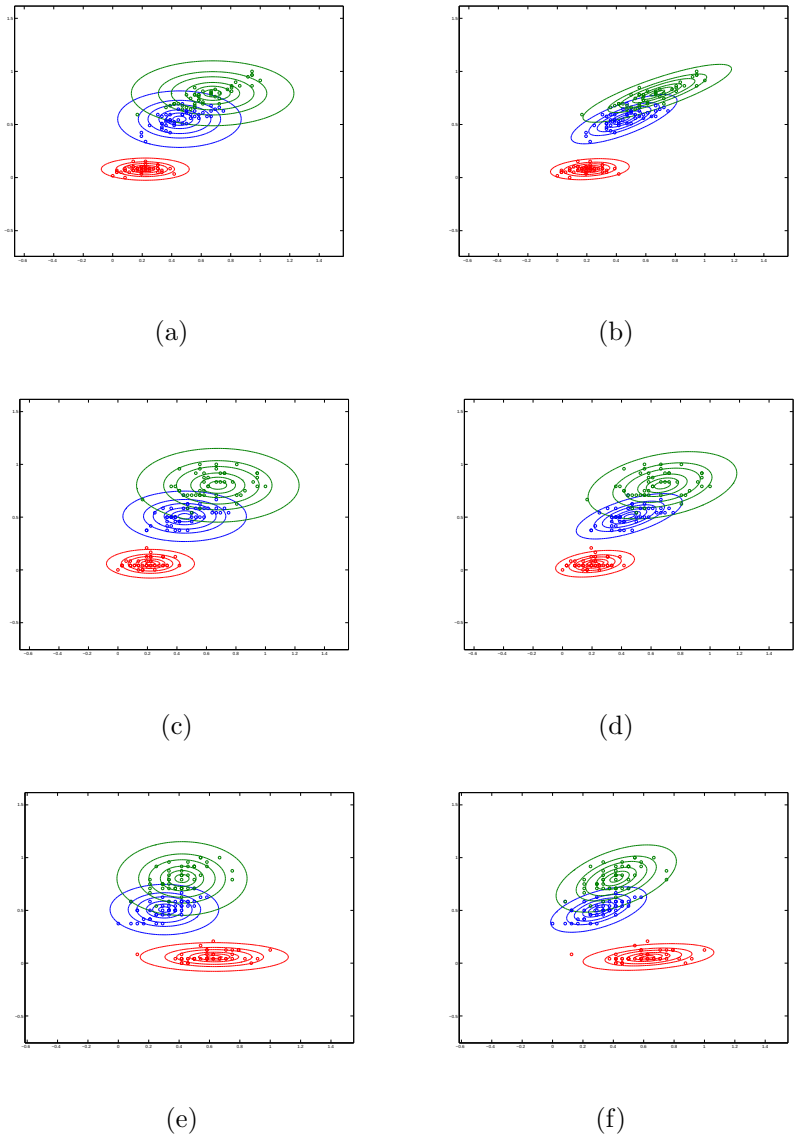
It must then be normalized as follows:

$$\bar{\beta}_i(\vec{x}) = \frac{\beta_i(\vec{x})}{\sum_{j=1}^{r} \beta_j(\vec{x})} \tag{3.6}$$

In our model, we go a step ahead in the structure of the antecedent part of TS models. In addition to the use of different variance values in the definition of the fuzzy prototypes in the input data space, we take into consideration the covariance between the features. Therefore, the fuzzy influence zone of each rule is represented by a prototype with a rotated hyper-elliptical form. For most of classification problems, especially with high dimensional input space, the covariance between the features cannot be ignored and it is difficult to cover the clouds of data points using axes-parallel hyper-elliptical prototypes. As a result, this improvement of the antecedent structure of TS models can play an important role in the overall achieved accuracy, as we will see in the results. The difference between the two types of antecedent is illustrated in figure 3.1 (data obtained from "Iris" dataset [55]).

Each fuzzy prototype in our system is yet represented by a center $\vec{\mu_i}$ and a covariance matrix $A_i$:

$$A_i = \begin{bmatrix} \sigma_1^2 & c_{12} & ... & c_{1n} \\ c_{21} & \sigma_2^2 & ... & c_{2n} \\ ... & ... & ... & ... \\ c_{n1} & c_{n2} & ... & \sigma_n^2 \end{bmatrix}_i \tag{3.7}$$



(a) (b)

(c) (d)

(e) (f)

Figure 3.1: The zone of influence of the prototype is parallel to the axes in (a), (c) and (e), while the rotated zones in (b), (d) and (f) result in more accurate data covering.

where $c_{12}(= c_{21})$ is the covariance between $x_1$ and $x_2$, and so on.

The three main points that must be solved in order to implement this enhanced antecedent structure in the evolving FIS are:

- how to measure the membership degree of a given input to the fuzzy prototype considering the variance/covariance matrix?
- how to incrementally (recursively) and efficiently calculate the variance/covariance matrix?

To address the first point, different multi-dimensional (multivariate) probability density functions can be used to measure the activation degree of each prototype. We can mention two of them:

- Multivariate normal distribution: the activation is calculated according to this distribution as follows

$$\beta_i(\vec{x}) = \frac{1}{(2\pi)^{n/2}\,|A_i|^{1/2}}\; exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu_i})^t A_i^{-1}(\vec{x}-\vec{\mu_i})\right) \tag{3.8}$$

- Multivariate Cauchy distribution: the activation here is defined as follows

$$\beta_i(\vec{x}) = \frac{1}{2\pi\sqrt{|A_i|}}\left[1+(\vec{x}-\vec{\mu_i})^t A_i^{-1}(\vec{x}-\vec{\mu_i})\right]^{-\frac{n+1}{2}} \tag{3.9}$$

After an experimental comparative study on different benchmark datasets, we found out that multivariate Cauchy distribution slightly outperforms the Normal distribution. However, the presented learning algorithm is independent of this choice, and the applied distribution has no effect on the manner of the estimation of the variance/covariance matrices.

The second point, about the recursive estimation of the variance/covariance matrices, will be answered in the next section under the name of "antecedent adaptation".

## 3.3 Evolve: an incremental learning algorithm for evolving TS classifiers

By analyzing the structure of TS FISs described in the Section 3.2, we note that it consists of a set of local linear sub-models represented by the linear hyper planes at

the consequent part of each rule. The zone of influence of each sub-model is defined by the antecedent of the fuzzy rule represented by a fuzzy prototype as described before. This structure can be illustrated as a three-layer neural network model like in Figure 3.2 (already presented in Section 2.5.2.3).

Any learning algorithm of an evolving TS FIS must address the following points:

- when and how new rules must be added to the system?
- how the antecedent part of the existing rules is adapted according to new samples?
- how the consequence parameters are updated according to new samples?
- how to guarantee a stable consequent learning in spite of the continuous antecedent adaptation?

The first three tasks must be done in an online incremental mode and all the needed calculation must be completely recursive. For the incremental rule creation, we use in our system a density-based incremental clustering method. Coupling this technique with the incremental adaptation of the antecedent of the rules is a part of the originality of our learning model. We use the recursive least squares method for estimating the linear consequent parameters in incremental manner.

Thus, "Evolve" learning algorithm consists of three distinct parts: rule creation (incremental clustering), antecedent adaptation and consequences learning. These three components are represented in Figure 3.2 by three different blocks. We describe in this section these three components in details, and we summarize the complete learning algorithm at the end of the section.

We note from Figure 3.2 that the input signal of the linear consequent learning process is the input vector $\vec{x}$ weighted by the vector of prototype activations. The modifications applied on the antecedent structure results in perturbations in the optimization of the parameters of the linear consequent functions.

The stability of consequent learning is one of the principal issues that need to be addressed. Although the enhanced antecedent structure presented in Section 3.2 offers more precise modeling of the clouds of data and thus improves the overall performance, the antecedent adaptation represented by the shifting of the centers of

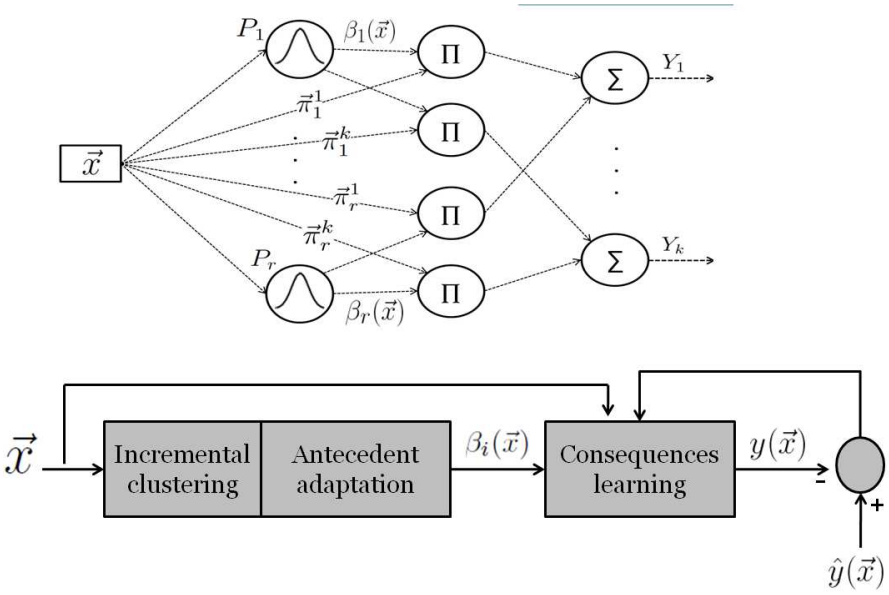



Figure 3.2: The different components of Evolve algorithm

the prototypes and updating their zones of influence produces an important perturbation on the consequent learning. This problem will be discussed in more details at the last of this section.

### 3.3.1 Density-based incremental clustering

As mentioned in the precedent chapter, the incremental clustering methods can be classified into distance-based methods and density-based methods. The major shortcoming in distance-based methods is the setting of the vigilance threshold that is a problem-dependent parameter. For this reason, we use in our algorithm a density-based method that does not depend on a threshold distance: the eClustering method.

In incremental clustering methods, each new incoming point will either belong to an existing cluster of points, or a new cluster will be detected and created. In distance-based incremental clustering methods, a new cluster is created if the new incoming point is sufficiently far from the existing focal points. In addition to the difficulty of setting of the threshold value that defines the minimum distance to create new cluster, these methods are sensitive to outliers and useless clusters may be created. In density-based methods, a new cluster is created only if the new incoming point sufficiently increases the density of its local region in the data space.

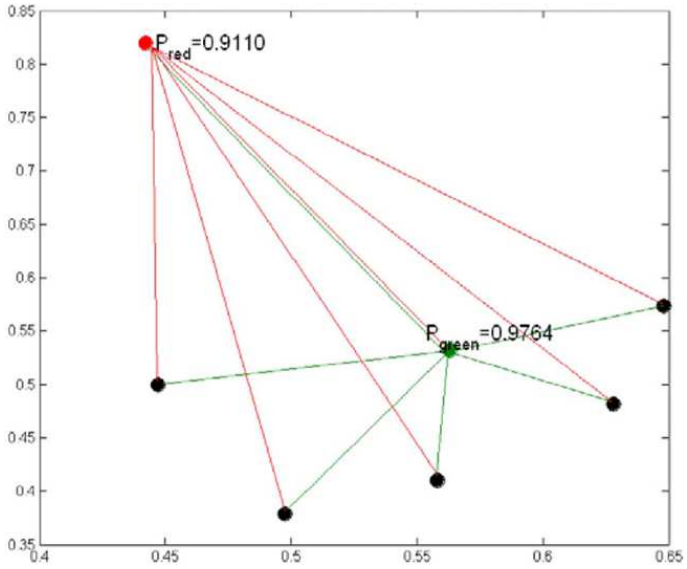



Figure 3.3: An example to illustrate the concept of the potential of data points [9].

The minimum density value to create a new cluster can be relative to the density of other clusters, as in eClustering, and no predefined threshold is needed. The eClustring method had been briefly presented in the precedent chapter. We give below a more detailed explanation about this method and the manner in which it is used in our system.

Yager and Filev proposed in [35] a batch (non-incremental) density-based clustering method, called "mountain clustering". In mountain clustering, a specific value, called "potential", is calculated for each data point as the inverse of the sum of distances between this point and all the other points (see figure 3.3):

$$Pot(\vec{x}_i) = \frac{1}{1 + (s-1)^{-1} \sum_{j=1, j \neq i}^{s} \|\vec{x}_i - \vec{x}_j\|^2} \tag{3.10}$$

where $s$ represents the size of the dataset.

The algorithm of mountain clustering can be briefly summarized as follows:

- Step 1: the point with the highest potential value is chosen as a cluster center.
- Step 2: for the rest, the potential value of each point is reduced in a manner inversely proportional to the distance between this point and the selected center.
- Step 3: go to step 1 if the number of desired clusters is not yet reached, go to end otherwise.

In order to adapt the potential concept to be used in an incremental clustering, we need a recursive formula that can be used to calculate the potential of a new incoming point $\vec{x}_t$ without requiring access to the $t-1$ precedent points as in the original formula. Angelov has deduced in [36] a recursive potential calculation starting from the original formula (Equation 3.10) that can be rewritten as follows:

$$Pot(\vec{x}_t) = \frac{1}{1 + (t-1)^{-1} \sum_{i=1}^{t-1} \sum_{j=1}^{n} (x_t^j - x_i^j)^2}$$

$$Pot(\vec{x}_t) = \frac{(t-1)}{(t-1) + \sum_{i=1}^{t-1} \sum_{j=1}^{n} \{(x_t^j)^2 - 2x_t^j x_i^j + (x_i^j)^2\}}$$

$$Pot(\vec{x}_t) = \frac{(t-1)}{(t-1) + (t-1)\sum_{j=1}^{n}(x_t^j)^2 + \sum_{i=1}^{t-1}\sum_{j=1}^{n}(x_i^j)^2 - 2\sum_{i=1}^{t-1}\sum_{j=1}^{n}(x_t^j)(x_i^j)}$$

For simplification, we define the following terms:

$$\alpha(t) = \sum_{j=1}^{n}(x_t^j)^2 \quad , \quad \gamma(t) = \sum_{i=1}^{t-1}\sum_{j=1}^{n}(x_i^j)^2 \quad , \quad \zeta(t) = \sum_{i=1}^{t-1}\sum_{j=1}^{n}(x_t^j)(x_i^j)$$

Using these terms, the potential equation for a new incoming point $\vec{x}_t$ can be rewritten as follows:

$$Pot(\vec{x}_t) = \frac{(t-1)}{(t-1) + (t-1)\alpha(t) + \gamma(t) - 2\zeta(t)} \tag{3.11}$$

The calculation of $\alpha(t)$ is straightforward. $\gamma(t)$ can be recursively estimated as follows :

$$\gamma(t) = \gamma(t-1) + \alpha(t-1), \quad \gamma(1) = 0$$

We can rewrite $\zeta(t)$ as follows:

$$\zeta(t) = \sum_{j=1}^{n}\left(x_t^j \,.\, \sum_{i=1}^{t-1}(x_i^j)\right)$$

Thus, $\zeta(t)$ can be recursively estimated using the following recurrence equations:

$$\zeta(t) = \sum_{j=1}^{n} x_t^j \, \eta^j(t) \quad , \quad \eta^j(t) = \eta^j(t-1) + x_{t-1}^j \quad , \quad \eta^j(1) = 0$$

In the same way, the potential of an existing focal point $\vec{\mu_i}$ is recursively updated according to incoming point using the next equation:

$$Pot(\vec{\mu_i}) = \frac{(t-1)Pot(\vec{\mu_i})}{(t-2) + Pot(\vec{\mu_i}) + Pot(\vec{\mu_i}) \sum_{j=1}^{n} \|\mu_i - x_{t-1}\|_j^2} \tag{3.12}$$

The mathematical reasoning of this recursive equation staring from the original one is given in [36].

The eClustering algorithm can then be summarized as follows:

- Step 1: The first sample is considered as the focal point and it potential is initialized by 1.
- Step 2: For each new incoming sample, its potential is calculated using Equation 3.11.
- Step 3: The potential of the existing focal points is updated by Equation 3.12.
- Step 4: If the potential of the new point is comparable to the potential of the existing focal points, it is considered as new focal point and its potential is initialized by 1.
- Go to Step 2.

This incremental clustering method is used in our learning algorithm in order to detect the necessity of adding new rules to the FIS. The data point $\vec{x_t}$ that gets a high potential and triggers a new rule creation will be considered as the center of a new prototype ($\vec{\mu_{r+1}} = \vec{x_t}$). An initial diagonal variance/covariance matrix is associated to the new prototype; $A_{r+1} = \epsilon I$, where $I$ is the identity matrix of size $n$ and $\epsilon$ is a problem-independent parameter and can be generally set to $10^{-2}$.

Therefore, a new fuzzy rule is added to the system so that its antecedent part is represented by the new prototype, and we will see later in this chapter how the consequence parameters are initialized.

### 3.3.2 Antecedent adaptation

As it can be noted in the previous section, the condition of having a high potential is hard and inversely proportional to the growing number of data. Thus, we can have a cluster center which is not really in the optimal position according to the data

history, but it keeps being center because it still has the highest potential value. Therefore, incremental clustering process of the antecedent part of the FIS will not be able to take advantage of the data points that do not have a very high potential to move (or reshape) the existing clusters.

We enhance the incremental clustering process by an adaptation algorithm which allows to incrementally update the prototype center coordinations according to each new available learning data, and to recursively compute the prototype covariance matrices in order to give them the rotated hyper-elliptical form. For each new sample $\vec{x}_t$, the center and the covariance matrix of the prototype that has the highest activation degree are updated.

The recursive estimation of the center can be found as follows:

$$\begin{aligned} \vec{\mu_t} &= \frac{1}{t}\sum_{i=1}^{t}\vec{x}_i \\ &= \frac{1}{t}(\sum_{i=1}^{t-1}\vec{x}_i + \vec{x}_t) \\ &= \frac{1}{t}((t-1)\vec{\mu}_{t-1} + \vec{x}_t) \\ &= \frac{t-1}{t}\ \vec{\mu}_{t-1} + \frac{1}{t}\ \vec{x}_t \end{aligned} \tag{3.13}$$

where $t$ represents the number of updates that have been applied so far on this prototype.

As mentioned in the last section, when the potential of the new sample is higher than the potential of the existing centers this sample will be a center of a new prototype, and the covariance matrix that defines the influence zone will be initialized by a matrix proportional to identity matrix (which results in a hyper-spherical shape).

In order to represent the repartition of the data cloud represented by the prototype, the shape of the prototype (its covariance matrix) must be adapted and updated according to each new data sample associated to this protoype.

We can prove that the covariance matrix can be recursively computed as follows:

$$A_t = \frac{t-1}{t}\ A_{t-1} + \frac{1}{t}\ (\vec{x}_t - \vec{\mu}_t)(\vec{x}_t - \vec{\mu}_t)^T \tag{3.14}$$

For practical issues, since the membership degree is calculated using $A^{-1}(|A| = \frac{1}{|A^{-1}|})$, and in order to avoid any matrix inversion, we use an updating rule for $A^{-1}$ directly:

$$A_t^{-1} = \frac{t}{t-1} A_{t-1}^{-1} - \frac{1}{t-1} \cdot \frac{(A_{t-1}^{-1}(\vec{x}_t - \vec{\mu}_i)) \cdot (A_{t-1}^{-1}(\vec{x}_t - \vec{\mu}_i))^T}{1 + \frac{1}{t}((\vec{x}_t - \vec{\mu}_i)^T A_{t-1}^{-1}(\vec{x}_t - \vec{\mu}_i))} \tag{3.15}$$

It is worth mentioning that Equations 3.13 and 3.14 are used in stationary incremental learning problems, in which all samples have the same importance regardless their time of arrival; i.e. later samples are as important as first ones. We consider our evolving handwriting classifier as a stationary problem, so that old data are as important as new ones.

For non-stationary problems, where the recent samples have more importance than old ones, the term $\frac{1}{t}$ is replaced by a constant value $a$.

### 3.3.3 Consequent learning

The coefficients of the consequent functions in our FIS must be adjusted after each new incoming data point. As we can understand from figure 3.2, the input vector $\vec{x}$ is weighted by the activation of each rule and used to optimize the consequence parameters of this rule in supervised manner (considering that the ground truth output vector $\vec{\hat{y}}(\vec{x})$ is available).

The consequent parameters of the fuzzy rules can be learned by global or local optimization. In the former, the consequences of all the rules are learned together and the learning aims at minimizing the global error achieved by the entire system. Whereas the consequences of the fuzzy rules in the latter is optimized separately and the error signal is measured on each rule and used to learn its consequent parameters.

In the rest of this section, we present first the global incremental learning of the consequent parameters. Then, the local learning formulas are as well presented. The application of the learning algorithm on the special case of zero-order TS FIS is briefly discussed at the end of the section.

#### 3.3.3.1 Global learning of consequent parameters

The problem of coefficients estimation of the linear consequences functions of a FIS can be seen as the problem of resolving a system of linear equations expressed as follows:

$$\Psi_i \, \Pi \; = \; Y_i \qquad i = 1, 2, ..., t \tag{3.16}$$

where $\Pi$ is the matrix of all the parameters of the linear consequences of the system.

$$\Pi = \begin{bmatrix} \vec{\pi}_1^1 & \vec{\pi}_1^2 & ... & \vec{\pi}_1^m \\ \vec{\pi}_2^1 & \vec{\pi}_2^2 & ... & \vec{\pi}_2^m \\ ... & ... & ... & ... \\ \vec{\pi}_r^1 & \vec{\pi}_r^2 & ... & \vec{\pi}_r^m \end{bmatrix}$$

$m$ represents the number of classes, $r$ is the number of fuzzy rules,
$\Psi_i = [\beta_1(\vec{x}_i)\vec{x}_i, \beta_2(\vec{x}_i)\vec{x}_i, ..., \beta_r(\vec{x}_i)\vec{x}_i]$ is the input vector weighted by the activation degrees of the prototypes, and $Y_i$ is the ground truth output vector.

Solving this system of linear equations by the least squares method consiste in minimizing the next cost function:

$$E = \sum_{i=1}^{t} \|\Psi_i \ \Pi \ - \ Y_i\|^2 \tag{3.17}$$

When using linear regression to solve physical or industrial problems, the input signal may contain a noise. In order to stabilize the solving of the system and to smooth the found solution, a regularization term (known as Tychonoff regularization) is added to the equation. The system is then expressed as follows:

$$(\Psi_i \ + \delta I)\Pi \ = \ Y_i \qquad i = 1, 2, ..., t \tag{3.18}$$

We can rewrite the cost function as follows:

$$E = \sum_{i=1}^{t} \|\Psi_i \ \Pi \ - \ Y_i\|^2 + \omega \|\Pi\|^2 \tag{3.19}$$

where $\omega = \delta^2$ is a positive number called the regularization parameter, and $I$ is the identity matrix.

The regularization parameter is important in our FIS because the antecedent adaptation produces a noise on the input $\Psi$ (We may have $\Psi_{t1} \neq \Psi_{t2}$ for $\vec{x}_{t1} = \vec{x}_{t2}$ because of the evolution of the antecedents).

The solution that minimizes the cost fonction of Equation 3.19 is:

$$\Pi_t = (\sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I)^{-1} \ . \ \sum_{i=1}^{t} \Psi_i Y_i \tag{3.20}$$

We rewrite Equation 3.20 by replacing $(\sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I)$ and $(\sum_{i=1}^{t} \Psi_i Y_i)$ by $\Phi_t$ et $Z_t$, respectively:

$$\Pi_t = \Phi_t^{-1} \ . Z_t \tag{3.21}$$

By isolating the term corresponding to $i = t$, we can rewrite $\Phi_t$ as follows:

$$\Phi_t = \left[\sum_{i=1}^{t-1} \Psi_i \Psi_i^T + \omega I\right] + \Psi_t \Psi_t^T \tag{3.22}$$

Thus, we can update the matrix $\Phi$ using the following recursive formula:

$$\Phi_t = \Phi_{t-1} + \Psi_t \Psi_t^T \tag{3.23}$$

In the same way, we can deduce a recursive formula to update the matrix $Z$ :

$$Z_t = Z_{t-1} + \Psi_t Y_t \tag{3.24}$$

In order to calculate $\Pi_t$ using Equation 3.21, we need to calculate the inverse of $\Phi$. In practice, we generally try to avoid the matrix inversion operation because it requires a lot of calculation time and may be the origin of instabilities. Moreover, we prefer to have a recursive equation for the calculation of $\Pi$ in incremental and efficient manner. We can realize these two goals thanks to the following lemma of matrix inversion:

**Lemma 1 :** Let $A = B^{-1} + CD^{-1}C^T$, we can write the inverse of $A$ as follows:

$$A^{-1} = B - BC(D + C^T BC)^{-1} C^T B \tag{3.25}$$

To apply Lemma 1 on Equation 3.23, we make the following substitutions:

$$A = \Phi_t, B^{-1} = \Phi_{t-1}, C = \Psi_t, D = 1$$

Thus, we can obtain the recursive formula of the inverse of the matrix $\Phi$:

$$\Phi_t^{-1} = \Phi_{t-1}^{-1} - \frac{\Phi_{t-1}^{-1}\ \Psi_t\ \Psi_t^T\ \Phi_{t-1}^{-1}}{1 + \Psi_t^T\ \Phi_{t-1}^{-1}\ \Psi_t} \tag{3.26}$$

Then, in order to obtain a recursive equation to calculate $\Pi$, we write:

$$\begin{aligned}
\Pi_t &= \Phi_t^{-1} Z_t = \Phi_t^{-1}(Z_{t-1} + \Psi_t Y_t) = \Phi_t^{-1}(\Phi_{t-1}\Pi_{t-1} + \Psi_t Y_t) \\
&= \Phi_t^{-1}((\Phi_t - \Psi_t \Psi_t^T)\Pi_{t-1} + \Psi_t Y_t) = \Pi_{t-1} - \Phi_t^{-1}\Psi_t \Psi_t^T \Pi_{t-1} + \Phi_t^{-1}\Psi_t Y_t \\
&= \Pi_{t-1} - \Phi_t^{-1}\ \Psi_t\ (Y_t - \Psi_t^T\ \Pi_{t-1})
\end{aligned} \tag{3.27}$$

The initialization of the algorithm consists in determine two quantities:

- $\Pi_0$ : In practice, and when no prior knowledge is available, $\Pi_0$ is initialized by 0.

- $\Phi_0^{-1}$ : Given $\Phi_t = \sum_{i=1}^{t} \Psi_i \Psi_i^T + \omega I$ and by putting $t$ equals 0, we find that $\Phi_0^{-1} = \omega^{-1} I$, where $\omega$ is the regularization parameter.

Large values of $\omega^{-1}$ (between $10^2$ and $10^4$) are generally adopted when the noise-to-signal ratio on the input vector is high, which is the case in our FIS especially in the beginning of the learning where significant modifications are done on the prototypes. The impact of the value of $\omega^{-1}$ on the performance of the algorithm according to the input noise level is discussed in [56].

When a new rule is added to the system, its parameters are initialized by the average of the parameters of the other rules:

$$\Pi_t = \begin{bmatrix} \vec{\pi}^1_{1(t-1)} & \vec{\pi}^2_{1(t-1)} & \dots & \vec{\pi}^m_{1(t-1)} \\ \vec{\pi}^1_{2(t-1)} & \vec{\pi}^2_{2(t-1)} & \dots & \vec{\pi}^m_{2(t-1)} \\ \dots & \dots & \dots & \dots \\ \vec{\pi}^1_{r(t-1)} & \vec{\pi}^2_{r(t-1)} & \dots & \vec{\pi}^m_{r(t-1)} \\ \vec{\pi}^1_{(r+1)t} & \vec{\pi}^2_{(r+1)t} & \dots & \vec{\pi}^m_{(r+1)} \end{bmatrix} \tag{3.28}$$

where

$$\vec{\pi}^c_{(r+1)t} = \sum_{i=1}^{r} \beta_i(\vec{x}_t) \vec{\pi}^c_{i(t-1)} \tag{3.29}$$

The matrix $\Phi^{-1}$ is extended as follows:

$$\Phi_t^{-1} = \begin{bmatrix} \rho \begin{bmatrix} \Phi_{t-1}^{-1} \end{bmatrix} & \begin{bmatrix} 0 \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} & \begin{bmatrix} \Omega^{-1} & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \Omega^{-1} \end{bmatrix} \end{bmatrix} \tag{3.30}$$

where $\rho = (r^2 + 1)/r^2$.

#### 3.3.3.2 Local learning of consequent parameters

Contrary to global learning, the local learning consists in estimating the consequences of each rule independently, so that the local error of each rule is considered and the optimization is carried out for each local region (defined by the rule antecedent). Similar to global learning, local learning can be performed in incremental mode by extending the weighted least squares method to a recursive formula.

Let $\Pi_i$ be the linear consequent parameters of the rule $i$:

$$\Pi_i = \left[\vec{\pi}_i^1 \;\; \vec{\pi}_i^2 \;\; ... \;\; \vec{\pi}_i^m\right]^T \tag{3.31}$$

The deduction of these formulas is quite similar to that presented in the precedent section. Therefore, the matrix of consequences parameters of each rule can be recursively estimated as follows:

$$\Pi_{i(t)} = \Pi_{i(t-1)} + C_{i(t)}\ \bar{\beta}_i(\vec{x}_t)\ \vec{x}_t\ (Y_t - \vec{x}_t\Pi_{i(t-1)}),\ \ \Pi_{init} = \vec{0} \tag{3.32}$$

$$C_i = C_i - \frac{\bar{\beta}_i(\vec{x}_k)C_i\vec{x}_k\vec{x}_k^tC_i}{1 + \bar{\beta}_i(\vec{x}_k)\vec{x}_k^tC_i\vec{x}_k},\ \ C_{init} = \Omega I \tag{3.33}$$

where $\Omega$ is a large positive number, and $I$ is the identity matrix.

It is important to mention that the main advantage of local learning is to avoid the large size of the matrix $\Phi$ when the number of rules in the systems increases. It will be demonstrated in the experimental results that the accuracy achieved by the system when local learning is used can be comparable to global learning performance. Therefore, local consequent learning must be preferred when the incremental learning is done in an online and interactive mode and a fast learning process is required.

#### 3.3.3.3 Learning of zero-order consequences

The solutions for incremental consequent learning (global or local) that we presented above can be applied in straightforward manner for either higher-degree linear functions or even for consequences of zero degree, that we call singleton consequences. Singleton consequences are used in the so called zero-order TS FISs as mentioned in the precedent chapter and result in less sophisticated models but with less precision. A fuzzy rule in zero-order TS FISs is defined as follows:

$$Rule_i : IF\ \vec{x}\ is\ P_i\ THEN\ y_i^1 = s_i^1\ AND \ldots AND\ y_i^k = s_i^k \tag{3.34}$$

To apply the RLS method on a zero-order TS FIS, the matrix of consequences parameters is written as follows:

$$S = \begin{bmatrix} s_1^1 & s_1^2 & ... & s_1^m \\ s_2^1 & s_2^2 & ... & s_2^m \\ ... & ... & ... & ... \\ s_r^1 & s_r^2 & ... & s_r^m \end{bmatrix} \tag{3.35}$$

where $m$ is the number of classes, and $r$ is the number of fuzzy rules.

Similar to first-order consequences, the matrix $S$ can be globally and incrementally estimated by:

$$S_k = S_{k-1} + C_k \psi_k (Y_k - \psi_k^T S_{k-1}) \; ; \quad S_1 = 0 \tag{3.36}$$

$$C_k = C_{k-1} - \frac{C_{k-1} \psi_k \psi_k^T C_{k-1}}{1 + \psi_k^T C_{k-1} \psi_k} \; ; \quad C_1 = \Omega I \tag{3.37}$$

Obviously, the low number of consequent parameters in zero-order TS models makes the learning process much faster compared to first-order models. However, it will be demonstrated in the experimental results that the performance of first-order models generally outperforms that of zero-order ones. The functional equivalence between zero-order TS FISs and radial basis function networks had been mentioned in the precedent chapter.

### 3.3.4 Learning stability using delayed antecedent adaptation

One of the favorable properties of the weighted recursive least squares method is that it converges to the optimal solution for each learning step. But, this property is only true when the weights (antecedent activations in our case) which are associated to the previous data points keep unchanged. In our system, the fuzzy prototypes that form the antecedent part are dynamic, they can be re-centered, shifted or reshaped according to the new data sample by the incremental clustering method (Section 3.3.1) and the adaptation algorithm (Section 3.3.2). Therefore, the older activations that have participated in the learning of the consequent parameters for the previous data are no longer the same. This makes the prior estimated linear consequent parameters non-optimal for the current antecedent model.

Although the existent evolving TS models like eTS and FlexFIS suffer from the same problem, the improvement of the antecedent structure in our model increases

this effect because the antecedent learning become more sophisticated and they will be continuously modified in a considerable manner.

In [7], the author points out this stability problem. He theoretically proposes to introduce, after each antecedent modification, a correction vector for the linear parameters and a correction matrix for their covariance matrix in order to balance out the current non-optimal solution toward the optimal one according to the degree of change in the premise part. To the best of our knowledge, there is not yet any proposed solution to estimate these correction vectors and matrices, it is still an open and sophisticated issue.

One first solution that we present to cope with the consequent learning stability problem in our FIS is to apply the antecedent adaptation in an incremental batch mode instead of a sample-wise mode. In this way, we keep on applying the consequent incremental learning algorithm in a sample-wise mode with fixed antecedent structure. In addition, we keep the data samples in a memory buffer with size $F$. Thus, after introducing $F$ data samples, the antecedent adaptation is applied for each sample in the buffer. Then, a readjustment of the consequent parameters is done by applying the RLS method on the samples in the buffer (with the modified antecedent structure). After this readjustment process, the buffer is emptied out and the antecedent structure "freezes" while the consequent parameters updating process continues in a sample-wise mode.

The advantage of this strategy is that the antecedent optimization is not scarified for the sake of stable consequent learning, but it is instead applied in temporal batches from time to time. However, as mentioned above this strategy requires a partial memory to temporary hold the data samples until the next antecedent adaptation.

The important role of using this delayed antecedent adaptation on the overall performance will be experimentally proved in the next chapter. However, the main drawback of this solution is the a-priori setting of the buffer size, which represents the only predefined parameter in the whole algorithm.

### 3.3.5 Evolve: the complet algorithm

The complete learning algorithm Evolve is summarized in Algorithm 1.

**Algorithm 1**: Evolve algorithm

```
Initialisation:
F, ω^{-1};
foreach new sample x⃗ do
    if x⃗ is the first sample of a new class then
        create a new fuzzy prototype based at x⃗;
        initialize its potential by 1;
        add a new fuzzy rule to the system;
        extend the consequent parameters matrix as in [3.28] and [3.29];
        update and extend the covariance matrix as in [3.30];
    else
        calculate the activations of the fuzzy rules by [3.9];
        determine the winning class label by [3.40];
        get the true class label;
        calculate the potential of x⃗ by [3.11];
        update the potentials of the existing prototypes centers using [3.12];
        if P(x⃗) > P_k(μ⃗_i)  ∀i ∈ [1, R] then
            create a new fuzzy prototype based at x⃗;
            initialize its potential by 1;
            add a new fuzzy rule to the system;
            extend the consequent parameters matrix as in [3.28] and [3.29];
            update and extend the covariance matrix as in [3.30];
        end
        update the consequents parameters using [3.36] and [3.37];
        add x⃗ to the memory buffer;
        if memory buffer is full then
            foreach sample x⃗_b in the buffer do
                Apply antecedent adaptation according to x⃗_b by [3.13] and [3.15];
            end
            foreach sample x⃗_b in the buffer do
                update the consequents parameters according to x⃗_b by [3.36];
            end
            dump the memory buffer;
        end
    end
end
```

## 3.4 A novel paradigm for stable incremental learning of TS models

It had been already mentioned that the main challenge in the incremental learning of evolving TS FIS is keeping as much as possible the coherence between the antecedent structure and the consequent parameters in order to get a high overall precision. Static TS models do not suffer from this problem because the antecedent structure is first learned using some clustering methods, then the consequent parameters are optimized while the antecedent structure keeps unchanged. This two-phase learning strategy cannot be used in an incremental learning of evolving TS models. As explained before, the learning of evolving TS models should be a lifelong process, and both the antecedent and the consequent part evolve according to new incoming data samples. The evolution of the antecedents is translated by either adding new prototypes or by adjusting the exiting prototypes. When a sophisticated antecedent structure is used to improve the system precision, the antecedent evolution becomes more important and the consequent learning process will be constantly perturbed because of that. For this reason, other approaches use often a simple antecedent structure with a limited evolution. In Evolve method, the learning of the enhanced antecedent structure is delayed and performed in incremental batch mode, using a partial memory buffer. This strategy provides some stability in the learning process and thus improves the overall performance.

However, and although the delayed antecedent adaptation represents a relatively efficient solution, it will be much better if some useless or not important antecedent modifications be completely avoided or minimized instead of being delayed. It can be noted from Section 3.3.2 that the re-estimation of prototypes' centers and covariance matrices takes place for each new incoming data points, and all the data points are considered in the same way. In order to minimize the antecedent modifications, we tried to find an intelligent strategy to distinguish between the dispensable and the indispensable antecedent modifications. The source of this information in this novel paradigm is the output error signal.

As mentioned in the precedent chapter, the ground truth label of each incoming data point is supposed to be available before using this point in the incremental learning process. The true label can be obtained in some applications by analyzing the reaction of the user on the label suggested by the system, which can be either

a direct or an indirect validation/correction action. More details about this process are given in the next chapter.

The true label is used to measure the error signal and then to supervise the consequent learning process (either global or local error is used as we have seen in the precedent section).

The fundamental idea of the novel approach is to use this error signal in the antecedent adaptation as well, contrary to existing approaches where the antecedent is learned independently from the consequent learning and the overall output.

The purpose of integrating the output signal in the antecedent learning is to bias it towards the incoming points with high output error; so that the more the error is high the more will be the influence of this point on the antecedent adaptation. The statistical antecedent adaptation used in Evolve algorithm does not take into consideration the output error committed for the incoming data point. All the data points are equally used in the adaptation recursive calculations regardless their output errors.

Thus, the main difference between the traditional learning paradigm used in Evolve and this novel paradigm is the fact that the antecedent adaptation is driven by an output error feedback as illustrated in Figure 3.4 (compared to Figure 3.2).

Thanks to this improvement, it becomes possible in the antecedent adaptation to focus on the data points that are misclassified by the system or correctly but hardly well-classified, and to put less focus on non-problematic points. This concept offers in same time two important advantages that can improve the overall performance of the evolving classifiers:

- reducing the antecedent modification for points with low output error, which enhances the stability of consequent learning,
- accelerating the learning by focusing on difficult points so that future misclassification errors can be avoided, especially at the beginning of the incremental learning process where only few learning data are available.

To formulate this concept, we rewrite the antecedent adaptation formulas (Section 3.3.2) as follows :

$$\vec{\mu_t} = \frac{t-w}{t} \ \vec{\mu}_{t-1} + \frac{w}{t} \ \vec{x}_t \tag{3.38}$$

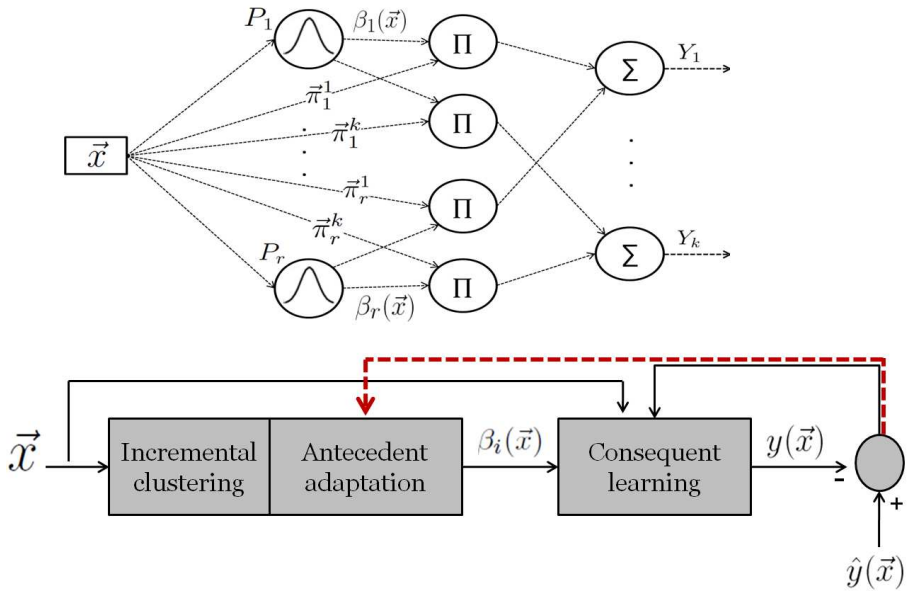



Figure 3.4: The new learning paradigm: antecedent adaptation driven by output feedback

$$A_t = \frac{t-w}{t} \; A_{t-1} + \frac{w}{t} \; (\vec{x}_t - \vec{\mu}_t)(\vec{x}_t - \vec{\mu}_t)^T \tag{3.39}$$

where $w$ represents the "weight" associated to the data point $\vec{x}_t$. The value of $w$ is related to the output error committed by the system for the current point $\vec{x}_t$.

We have studied two different manners for integrating the output error signal in the antecedent adaptation formulas. In the first one, the error is represented by the difference between the ground truth and the current output for each class. Whereas in the second method, the error is estimated by the difference between the score of the true class of the incoming point and the highest score between the other classes. This difference will be called the confusion degree of this point. These two strategies are presented in the next two sections.

### 3.4.1 Evolve+: antecedent learning based on global error feedback

It is useful to remind the reader that the output of our FIS is a vector of scores each of which corresponds to a specific class (a gesture in our application), and within the range [0,1]. Ideally, the score of the real class of $\vec{x}$ equals one, and the other

scores are zeros. This is how the ground truth output vector is set in the incremental learning of $\vec{x}$, while to recognize a new sample the winning class label is given by finding the maximal output and taking the corresponding class label as response:

$$class(\vec{x}) = y = argmax\ y_c(\vec{x}) \qquad c = 1, .., k \tag{3.40}$$

The global error is measure by the distance between the system output vector $\vec{y}$ and the ground truth vector $\hat{\vec{y}}$ (contains 1 for the true class and 0s for the rest). Thus, the value of $w$ can be estimated as follows:

$$w = \frac{1}{2} \sum_{c=1}^{k} \mid \hat{y}_c - y_c \mid \tag{3.41}$$

where $k$ is the number of classes.

The system outputs $y_c$ are normalized so that their sum equals 1. The value of $w$ is therefore between 0 and 1. Completely misclassified points will have a weight close to 1, while the weight of well-classified points will be close to 0.

Using this solution, the antecedent structure will not be modified to cover new incoming points if they are correctly classified and their output scores are close to ideal values (one for the true class, and zero for other classes). In the same time, the prototypes will move faster towards the data points that produce high output error, and the effect of these points on the covariance matrix will be reinforced. As explained above, the relative ignorance of "easy" points in the antecedent learning reduces the perturbation on consequent learning. The antecedent structure will not correspond to an exact representation of the distribution of data points in the feature space, as in Evolve method and the other existing approaches, like eTS, FlexFIS and DENFIS. The antecedent structure will be instead learned in harmony with the consequences. This is very important in an online incremental learning when the system performance must be improved as fast as possible using few data points. This new learning paradigm will be experimentally validated in the next chapter through the improvement on the accuracy of the system using Evolve+ algorithm, compared to Evolve algorithm.

Evolve+ is a general implementation of the novel learning paradigm, and can be used to incrementally learn any evolving TS FIS, independently from the type of application. Most of existing evolving approaches (eTS, FlexFIS, Denfes) had

been widely used in time-series modeling and prediction problems. In addition to classification problems, Evolve+ can perfectly replace the existing approaches for prediction problems. System performance in this type of problems is estimated by measuring the prediction error, which can be implemented using different quality measure, like R-squared adjusted [7], non-dimensional error index (NDEI) [5], or normalized RMSE [36].

For classification problems, the quality of the system is measure by its recognition rate. The output vector represents generally the scores for all the considered classes. The class with the largest score is considered as winner (Winner-take-all principal), regardless the absolute score value.

As explained before, the purpose of extending Evolve to Evolve+ is to reduce the antecedent modification by focusing on "difficult" points and ignoring "easy" ones. Following the same idea, the novel paradigm can be implemented for classification problem using a different weight estimation formula. The global error used in Equation 3.41 can be replaced in classification problems by a specific measure that represents the risk of misclassification associated to the input sample. This concept represents the core idea of Evolve++, a different implementation of the novel learning paradigm, dedicated for evolving classification problems.

### 3.4.2 Evolve++: confusion-driven antecedent learning

As mentioned in the precedent section, the system quality in time-series prediction problems is generally related to the difference (the distance) between system output vector and real output vector, and can be measures using different indications. In classification problem, the quality is perceived by the user from the number of misclassification errors committed by the system.

We have seen that the principal of the proposed learning paradigm is to give a special priority in the antecedent learning process for some data points, and less priority for others, in order to get stable and high overall performance. This priority is calculated in Evolve+ based on the global output error committed by the system for the given input vector (Equation 3.41). In the case of classification problems, where the purpose is to avoid misclassification errors, the priority can be calculated so that it becomes higher for points with a high risk of misclassification, regardless their global output error. Obviously, minimizing the global error will lead to less misclassification, but the idea of Evolve++ is to reduce the weights of points with

relatively low risk of misclassification, even if their global output error is still considerable. In this manner, we reduce the antecedent modifications that may not have a clear and direct impact on the system quality as perceived by the user in classification problems, which results in more stable consequent learning and more efficient antecedent structure. By focusing on the samples with high risk of misclassification, the improvement of the recognition rate of the classifier will goes faster during the incremental learning process, which is very important for the user especially at the beginning of the learning process or when adding new classes.

In Evolve++, we estimate the risk of misclassification of each sample by its confusion degree. The confusion degree is inversely proportional to the difference between the score of the true class of $\vec{x}$, and the highest score within the other (wrong) classes. The confusion-driven antecedent learning in Evolve++ is then implemented by calculating the weight $w$ of each incoming sample as follows:

$$w = (1 - [y_{\hat{c}} - y_{nc}])/2 \qquad w \in [0, 1] \tag{3.42}$$

where $y_{\hat{c}}$ is the system output corresponding to $\hat{c}$ that represents the true class of $\vec{x}_t$, and

$$y_{nc} = argmax \;\; y_c \;\; ; \qquad c = 1...k \;\; \& \;\; c \neq \hat{c} \tag{3.43}$$

Similar to Equation 3.41, the value of $w$ tends toward 0 when $\vec{x}_t$ is "strongly" recognized, and toward 1 when it is misrecognized. The risk of misclassification associated to $\vec{x}_t$ is proportional to the confusion degree estimated by the value of $w$.

Therefore, measuring the confusion degree in Evolve++ instead of the overall output error used in Evolve+ improves the discrimination capacity of the system and reduces the misclassification rate. By focusing the learning on data points with high misclassification risk, potential future misclassification errors can be avoided. This idea is translated by giving a relatively high weight for the corresponding data points in the antecedent adaptation. We present below a simple case study to illustrate the difference between the two methods.

In Table 3.4.2, let $\vec{x}_{t1}, \vec{x}_{t3}, \vec{x}_{t3}, \vec{x}_{t4}$ four data points from the same class $\hat{c}$, $y_{\hat{c}}$ the output score for this class, $y_{nc(i)\;\; i=1...5}$ the scores of the other classe, $w_1$ the weight associated to the point in Evolve+ based on global error (Equation 3.41), and $w_2$ its weight in Evolve++ based on the confusion degree (Equation 3.42).

| | $y_{\hat{c}}$ | $y_{nc1}$ | $y_{nc2}$ | $y_{nc3}$ | $y_{nc4}$ | $y_{nc5}$ | $w_1$ | $w_2$ |
|---|---|---|---|---|---|---|---|---|
| $\vec{x}_{t1}$ | 0.5 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.5 | 0.3 |
| $\vec{x}_{t2}$ | 0.5 | 0.5 | 0 | 0 | 0 | 0 | 0.5 | 0.5 |
| $\vec{x}_{t3}$ | 0.3 | 0.3 | 0.1 | 0.1 | 0.1 | 0.1 | 0.7 | 0.5 |
| $\vec{x}_{t4}$ | 0.3 | 0.7 | 0 | 0 | 0 | 0 | 0.7 | 0.7 |

Table 3.1: A case study to illustrate the difference in weighting strategies between Evolve+ and Evolve++

Although $\vec{x}_{t1}$ and $\vec{x}_{t2}$ have the same true class score (0.5), we note clearly that there is more risk of misclassification for $\vec{x}_{t2}$ than $\vec{x}_{t1}$. The global error strategy of Evolve+ gives these two samples the same weight in the antecedent adaptation, while the confusion-driven strategy in Evolve++ gives more importance for the learning $\vec{x}_{t2}$; i.e. $\vec{x}_{t2}$ will have more impact on the adaptation than $\vec{x}_{t1}$. The same conclusion can be made for $\vec{x}_{t3}$ and $\vec{x}_{t4}$ that are not correctly classified by the system.

We present in the next chapter an experimental demonstration of the superiority of Evolve++ compare to Evolve+, in evolving classification problems.

## 3.5 Open issues

We discuss in this section two important topics related to incremental learning. The first one concerns incremental unlearning (or forgetting) strategies that can be sometimes used in the evolving approaches depending on the applicative context. The second important topic is related to the potential effect of data ordering on the performance of the incremental algorithms. A brief analysis of the sensitivity of our algorithm to this effect is presented. However, it is worth mentioning that these two aspects represent interesting future tracks that require more in-depth investigation.

### 3.5.1 Forgetting strategy

One of the interesting issues in evolving approaches is their capacity, if any, of forgetting and unlearning old data. In many application areas, significant drifts in the characteristics of the physical system may occur, and an efficient forgetting strategy must be used in the evolving system to allow faster adjustment in such

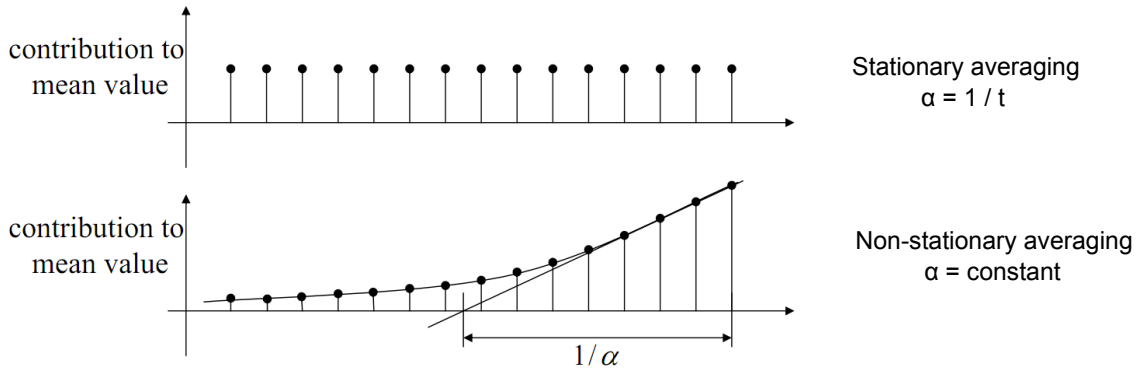



Figure 3.5: The difference between stationary and non-stationary averaging.

cases. This kind of problems, where new data is more significant than old one, is sometimes referred to as "non-stationary".

In the context of evolving handwriting recognition systems, the problem is supposed to be "stationary". Although new classes can be added, but there is generally no significant drifts in the nature of the learned classes. Given that the recognition system is designed to be used by the same user, there is no reason for radical and brutal modification in the learned gestures that may make old instances invalids and must be forgotten. Generally, the knowledge extracted from new samples from an existing class is accumulated to the existing knowledge in an incremental stationary manner.

As mentioned in Section 3.3.2, the antecedent adaptation formulas of Evolve algorithm can be modified to cope with non-stationary problems by replacing the variable term $\frac{1}{t}$ by a constant value $\alpha \in [0,1]$, as in Equations 3.44 and 3.45. When a constat value $\alpha$ is used in the adaptation formulas, the newer samples will have a higher weight that the old ones, whereas all samples have the same weight independent on the time of occurrence when the term $\frac{1}{t}$ is used as in Equations 3.13 and 3.14, so that later samples are considered as important as first ones (Figure 3.5 illustrates this difference).

$$\vec{\mu_t} = (1-\alpha)\ \ \vec{\mu}_{t-1} + \alpha\ \ \vec{x}_t \tag{3.44}$$

$$A_t = (1-\alpha)\ \ A_{t-1} + \alpha\ \ (\vec{x}_t - \vec{\mu}_t)(\vec{x}_t - \vec{\mu}_t)^T \tag{3.45}$$

An exponential forgetting factor can be added to the cost function of the least

squares method (Equation 3.17) so that it can be rewritten as follows:

$$E = \sum_{i=1}^{t} \lambda^{t-i} \left\| \Psi_i \, \Pi \; - \; Y_i \right\|^2 \tag{3.46}$$

where $\lambda \in [0, 1]$.

Based on this cost function, the forgetting factor appears in the calculation of the matrix $\Phi_t^{-1}$ in the formulas of recursive least squares (Equation 3.26), as follows:

$$\Phi_t^{-1} = \frac{1}{\lambda} \left[ \Phi_{t-1}^{-1} - \frac{\Phi_{t-1}^{-1} \, \Psi_t \, \Psi_t^T \, \Phi_{t-1}^{-1}}{\lambda + \Psi_t^T \, \Phi_{t-1}^{-1} \, \Psi_t} \right] \tag{3.47}$$

Small values of $\lambda$ lead to small contribution of old samples. This makes the method more sensitive to recent samples. The $\lambda = 1$ case is referred to as the growing window RLS algorithm, so that all the samples have the same weight in the optimization process.

Although our problem is considered as quasi-stationary as mentioned above, the forgetting factor can be used in the consequent learning by RLS method in order to focus on recent samples when significant antecedent modifications occur. It might be an interesting perspective to overcome the instability problems in the consequent learning, and it can be compared to the solution proposed in Evolve+ and Evolve++. However, a dynamic value of $\lambda$ must be used in an intelligent manner; our preliminary experiments have not shown very good results using simple constant value of $\lambda$.

### 3.5.2 Ordering effect

It is known that most incremental learning approaches are order dependent. The performance of the system may demand on the order of data presentation, and different ordered sequences of these data lead to different learning results [10]. An order-independent incremental learning system must not be sensitive to this phenomenon. If it is incrementally learned using $n$ data point, it must give the same results at the end of the learning process for the $n!$ possible orders (Figure 3.6).

The consequent learning by RLS method is by definition order-independent, because it converges to the optimal solution after each data point. The statistical antecedent adaptation in Evolve is order-independent as well. However, it is obvious that these two calculations become order-dependent when a forgetting strategy is adopted.

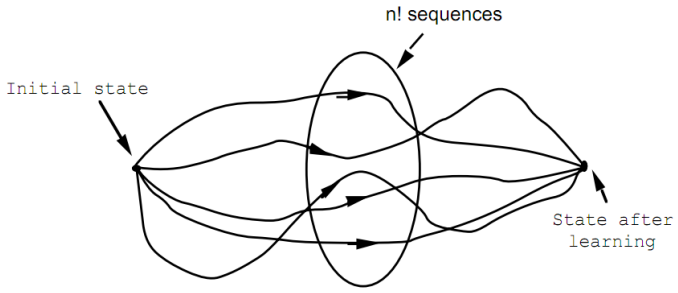



Figure 3.6: Order-independent incremental learning system [10]

The density-based incremental clustering method can be affected by the data ordering. The number of clusters ,i.e. the number of rules in our system, may vary according to the data ordering. Overcoming this drawback and thus making the whole system order-independent is one of the perspectives of this work.

## 3.6 Conclusion

In the context of dynamic classification problems, we have presented in this chapter an evolving classification system based on a Takagi-Sugeno fuzzy inference system. Contrary to other approaches, we define the antecedent structure using multidimensional membership functions, which leads to more precise data covering. We have proposed a learning algorithm, called Evolve, to incrementally train the system. It comprises a density-based incremental clustering technique, a prototype adaptation (shifting and reshaping) formulas and a consequent optimization using recursive least squares method. The stability issue in consequent learning had been addressed, and a partial memory-based solution had been proposed in Evolve to cope with this problem.

Furthermore, we have proposed a new learning paradigm that deals with the stability problem by integrating an output error feedback in the antecedent adaptation formulas. The role of this feedback signal is to avoid unnecessary antecedent modification and thus to reduce consequent perturbations. Therefore, the main advantages of the new paradigm consists in:

- moving the prototypes faster toward the data points that are not yet sufficiently learned by the system. The covariance matrix is also updated with more impact for these points than the others.

- avoiding the useless antecedent modification according to the points with low output error, which enhances the stability of consequent learning and improves the overall system precision.

Two learning algorithms that implement this new paradigm have been presented. The first one, Evolve+, considers the global output error and can be used for either classification or prediction problems and single-output or multi-outputs Takagi-Sugeno models. The second algorithm, Evolve++, only takes into consideration the risk of misclassification of the input data point, regardless the absolute committed error. On the one hand, the capacity of misclassification anticipation of Evolve++ helps at accelerating the learning, and, on the other hand, it offers more significant reduction of less important antecedent modifications and thus more stable consequent learning. Obviously, Evolve++ can only be valuable in classification problems with more than two classes.

In the next chapter, we present an experimental validation of the performance of the different incremental learning algorithm proposed in this chapter. A comparison with a well-known evolving approach will be presented, and several benchmark datasets will be considered in the tests.

# Chapter 4

# Experimental Validation

## 4.1 Introduction

We have presented in the precedent chapter an evolving classification approach based on a Takagi-Sugeno fuzzy inference system. We have proposed a new antecedent structure in order to improve the global system accuracy. An appropriate adaptation technique is used to incrementally learning the new antecedent structure, along with a well-known incremental clustering method. A complete incremental learning algorithm, called "Evolve", has been explained. In Evolve algorithm, the antecedent learning is carried out in an incremental batch mode in order to improve the stability of the consequent learning, which is performed using recursive least squares method.

Moreover, we have proposed a new learning paradigm that differs from the traditional one by the synergy between antecedent and consequent learning. The output error is used to control the antecedent learning process in order to get more stable and higher performance. Evolve algorithm has been extended to "Evolve+" according to the new learning paradigm. Additionally, the proposed learning paradigm is implemented in a second manner (based on confusion-driven antecedent learning) especially dedicated to classification problems. This third version of the learning algorithm has been called "Evolve++".

In this chapter, we present an experimental study in order to validate our propositions and to evaluate the performance of the different algorithms. The results of the proposed algorithms are compared to the results of a well-know evolving classification system called "eTS" (Section 2.5.3.5), and to two batch classification methods (K-NN and MLP). In this experimental evaluation, we aimed at covering the different points that has been mentioned in the precedent chapter, for example:

- the difference between global and local consequent learning,
- the difference between zero-order and first-order consequences,
- the impact of the enhanced antecedent structure, by comparing our system with Evolve algorithm to the well-known eTS system (a similar evolving TS model with a simpler antecedent structure),
- the impact of using the memory buffer in Evolve algorithm,
- the efficiency of the new learning paradigm, by comparing Evolve and Evolve+ algorithms,

- and the gain obtained thanks to Evolve++, a confusion-based implementation of the new learning paradigm, compared to Evolve+, a global error-based implementation.

The experiments are essentially carried out on a dataset of handwritten gestures, which represents the application area of our work. However, other well-known benchmark datasets are also used in some experiments in order to facilitate future comparisons.

The chapter is organized as follows: the datasets used in our experiments are presented in Section 4.2, then, we explain our experimental protocol in Section 4.3. The experimental results are presented in details in Section 4.4. Finally, we present in Section 4.5 the mechanism of integrating our evolving classifier in real applications. Three examples of these applications are briefly presented.

## 4.2 Classification datasets

It is useful to remind that the motivation of this research work is to find an evolving handwriting classifier that can be integrated in any pen-based application as we will see in Section 4.5. Therefore, we use in our tests a handwritten gesture dataset to prove the efficiency of the system before applying it in real applications.

Moreover, we test the model on several well-known classification benchmarks to validate the obtained results on larger scale and to allow other incremental approaches to compare with our model, in a dynamic classification context.

### 4.2.1 SIGN dataset

We led the experiments on "SIGN" dataset, which is a dataset of on-line handwritten gestures (Figure 4.1). It is composed of 25 different gestures drawn by 11 different writers on Tablet PCs. Each writer has drawn 100 samples of each gesture, *i.e.* 2,500 gestures in each writer-specific dataset. The data collection sessions were performed at Synchromedia laboratory, and Imadoc team. For any comparison purpose, the dataset (and additional information on the data collection protocol) can be freely downloaded [57].

In our experiments, each gesture is described by a set of 10 features. The presented results are the average of results of 11 different tests for the 11 writers.

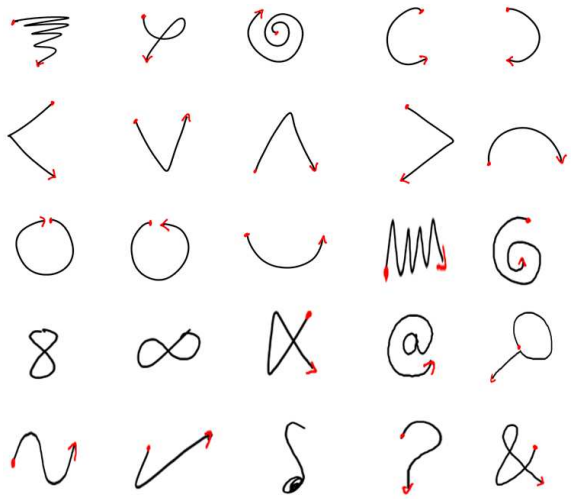


Figure 4.1: Handwritten gestures in the dataset SIGN

### 4.2.2 UCI datasets

Besides the SIGN dataset, we evaluate our algorithms on some benchmark problems form the UCI machine learning repository [55]. We followed two criteria in selecting the datasets :

- They should represent multi-class problems. Our learning algorithms (specially Evolve++) are optimized for classification problem with more than two classes. Thus, the two-class dataset are not preferred in our tests.
- The number of samples per class in each dataset should be large enough for two reasons. The first is to have a large learning dataset that allows continuing the incremental learning as far as possible and to examine the behavior of the algorithms in the long term. The second reason is to have a large test dataset to be able to correctly evaluate the classifier during the incremental learning process as we will see later. The large size of the test dataset helps as well to neutralize as much as possible the order effect on the results.

Respecting the last two criteria, we have chosen from UCI machine learning repository the next datasets to use them in our experiments:

- **CoverType**: The aim of this problem is to predict forest cover type from 12 cartographical variables. Seven classes of forest cover types are considered in this dataset. We use in our experiment a subset of 2100 instances.

| Dataset name | Nb. of classes | Nb. of features | Nb. of instances |
|---|---|---|---|
| Sign | 10 | 25 | 2500 |
| CoverType | 7 | 54 | 2100 |
| PenDigits | 10 | 16 | 10992 |
| Segment | 7 | 19 | 2310 |
| Letters | 26 | 16 | 20000 |

Table 4.1: Characteristics of learning datasets

- **PenDigits**: The objective is to classify the ten digits represented by their handwriting information from pressure sensitive tablet PC. Each digits is represented by 16 features. The dataset contains about 11000 instances.
- **Segment**: Each instance in the dataset represents a 3x3 region from 7 outdoor images. The aim is to find the image from which the region was taken. Each region is characterized by 19 numerical attributes. There are 2310 instances in the dataset.
- **Letters**: The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. Each letter is represented by 16 primitive numerical attributes. The dataset contains 20000 instances.

The characteristics of the five datasets are summarized in Table 4.1. The five dataset used in our experiments varies by both the number of features (number of data space dimensions), and the number of classes, which allows testing our algorithms on different multi-class problems.

## 4.3 Experimental protocol

Each learning dataset is split into a learning set and test set. The learning set represents in our experiments 75% of the entire dataset, and is used in the incremental learning process. The test set is used to estimate the performance during and at the end of the learning process. The learning/test protocol that we used in our experiments is shown in Figure 4.2(a). The instances of the learning dataset are sequentially introduced to the system in sample-wise mode. However, we used the

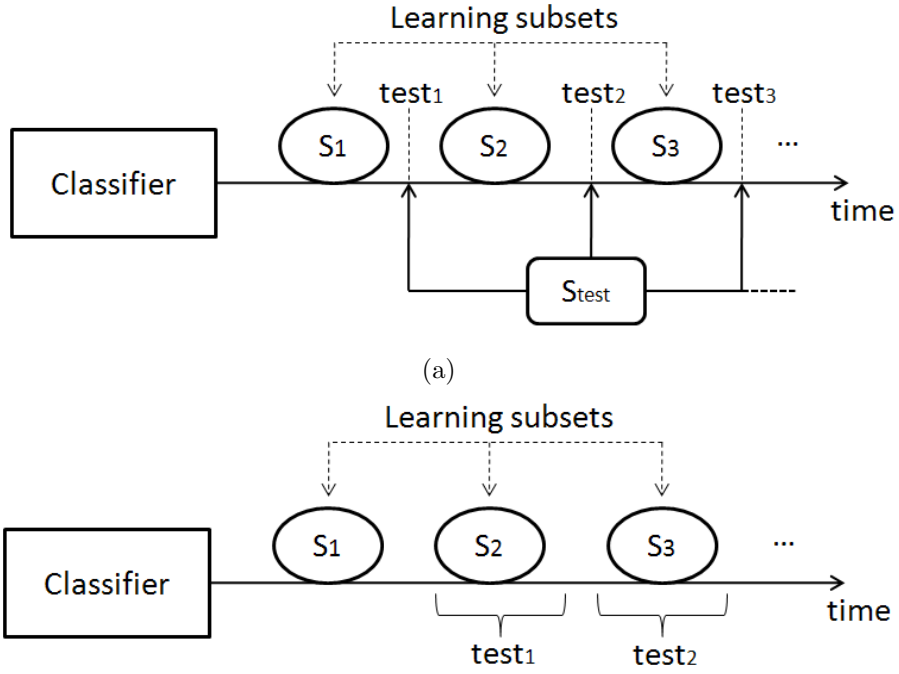



Figure 4.2: Learning/test protocols for incremental learning problems

term "learning subset" to identify the group of instances learned by the system between each two consecutive tests. Each learning subset $S_i$ contains the same number of instances per class.

An alternative learning/test protocol is shown in Figure 4.2(b). It can be noted from Algorithm 1 in the precedent chapter that a forward recognition step is performed for each learning instance. Although this step is required by the incremental learning process to get the output error, it can be further used to evaluate the classifier performance. There is no need in this second protocol for an independent test dataset $S_{test}$ as for the first one, which makes it suitable when only a small amount of data samples is available. It is useful to mention that the obtained recognition rate does not correspond to the traditional learning error rate mentioned sometimes in classic learning problems, which represents the error rate for the learning dataset. This is because the recognition of a new incoming sample is considered before including it in the learning process, as mentioned in Algorithm 1.

When the misclassification rate obtained on $S_{i+1}$ is used to evaluate the performance of the classifier after the learning of $S_i$, the evolution of the classifier due to the learning of $S_{i+1}$ is neglected. This neglecting may produce biased results when

the size of $S_{i+1}$ is considerable. The best solution is to let the size of each subset equals one, so that the performance of the classifier after the learning of sample $x_t$ is measured by the capacity of recognizing $x_{t+1}$. Thus, the experiments must be sufficiently repeated in order to estimate the misclassification rate, similar to the principal of Leave-one-out evaluation protocol.

In order to get the results unbiased by the data order effect, we repeat the experiment for each dataset 40 times with different random data orders and the mean results and standard deviations are presented in the figures and the tables.

We show in all the results the recognition rate achieved on each dataset using two traditional batch classification methods. The first one is based on a multi-layer perceptron with one hidden layer, and the second is a K-nearest neighbor with optimized value of k. The well-know WEKA workbench [58] had been used to perform the tests related to these two traditional classification techniques.

The main difficulty that we faced in our experiments is the comparison with other evolving approaches. Most of them have been tested on problems related to dynamic time-series prediction and modeling in different fields like control, diagnostic, etc. A small number of existing evolving approaches have been used in multi-class classification problems. The second difficulty is the absence of incremental learning competition or standard evaluation scenarios. The benchmark datasets must be associated with specific incremental learning protocols in order to allow comparison between approaches. Some works about evolving approaches that have been tested on benchmark data present only the final obtained recognition rate with no information about intermediate results [9] [59]. For these reasons, implementing other evolving approaches becomes mandatory in order to be able to compare with their results.

In our experimental study, we have chosen to implement a well-known evolving system called "eTS" [36], and to use it as a reference model. As mentioned in Section 2.5.3.5, eTS model is based on an evolving first-order Takagi-Sugeno model. The antecedent structure in eTS is based on axes-parallel hyper-elliptical prototypes. The spreads of the membership functions are recursively calculated using the recursive variance estimation formulas.

Different publications have proved the superiority of eTS over other evolving systems like DENFIS and FLEXFIS approaches [9] [59].

The main difference between eTS and Evolve is the antecedent structure. The

univariate membership functions used in eTS are replaced by multivariate ones in Evolve, and a variance/covariance matrix is recursively estimated for each prototype.

## 4.4 Experimental results

### 4.4.1 Global evaluation of the different learning algorithms

We have used the first experimental protocol explained in the precedent Section (Figure 4.2 (a)). The size of the learning subsets for each experiment is given in Table 4.2.

| Dataset name | $\|S_i\|$ | Nb. of samples per class in $S_i$ | $\sum_i \|S_i\|$ | $\|S_{test}\|$ |
|---|---|---|---|---|
| Sign | 50 | 2 | 1500 | 1000 |
| CoverType | 56 | 8 | 1470 | 630 |
| PenDigits | 360 | 36 | 6700 | 4300 |
| Segment | 77 | 11 | 1785 | 525 |

Table 4.2: The size of the learning subsets and the test dataset in each experiment

We present first in Figure 4.3 the results of a global experiment carried out on the handwritten gesture dataset, Sign, in order to give an exhaustive comparison between the different algorithms presented in the precedent chapter. Five evolving approaches are compared in this experiment:

- eTS: a state-of-the-art evolving system used as reference model in our results (described in Section 2.5.3.5).
- Evolve-noBuffer: our evolving TS system with enhanced antecedent structure, but no memory buffer is used in the incremental learning algorithm and the antecedent adaptation is performed after each new incoming sample. (see Section 3.3.4)
- Evolve: our evolving TS model with Evolve algorithm. The use of delayed antecedent adaptation based on a memory buffer makes the difference between Evolve and Evolve-noBuffer. Applying the antecedent adaptation in partial temporal batches is supposed to result in a more stable consequent learning and thus a higher recognition performance. The size of the memory buffer is empirically set to 30 samples in this experiment. (see Section 3.3.4)

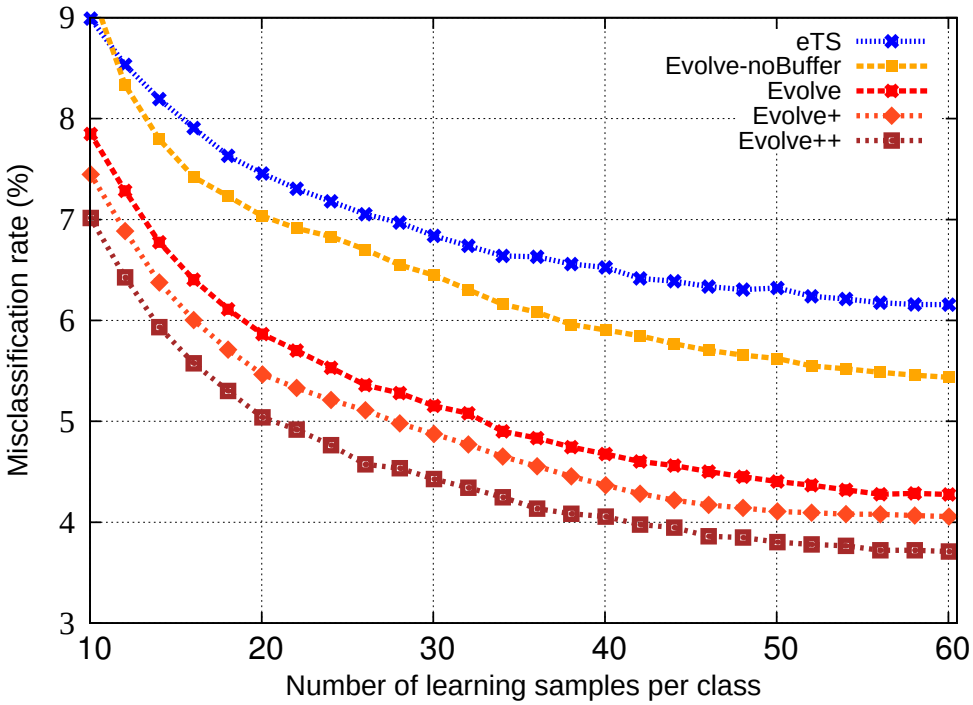



Figure 4.3: Evolution of classification performance during the incremental learning process (SIGN)

- Evolve+: a first implementation of the new global learning paradigm that we have proposed in which the antecedent adaptation is guided (and limited) by the output error feedback. This error is estimated in Evolve+ as the global absolute error (see Section 3.4.1).

- Evolve++: a second implementation of the new learning paradigm. It is based on a confusion-driven antecedent adaptation and supposed to be more suitable for the special case of multi-class classification problems than Evolve+. As explained in Section 3.4.2, Evolve++ aims at reducing the number of misclassification errors instead of minimizing the global error as in Evolve+.

We note globally from Figure 4.3 that our four models outperform (by different degrees) the eTS model thanks in the first place to the antecedent structure with multivariate membership functions.

However, the continuous perturbation on consequent learning produced by the sample-wise antecedent adaptation results in less efficient consequent and thus less overall recognition rate, in spite of the enhanced antecedent structure. This negative effect can be noted from Figure 4.3 by comparing Evolve-noBuffer results with

Evolve results. We note that the technique proposed in Evolve to deal with the instability in consequent learning offers a considerable amelioration in the system performance.

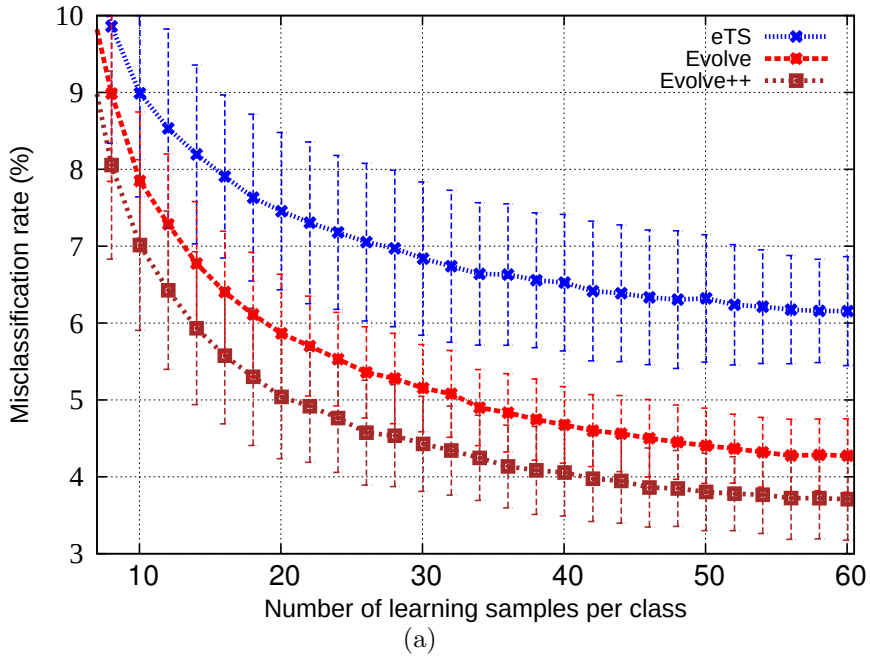
On the other hand, we can notice from the same figure that the algorithms of our proposed global learning paradigm (Evolve+ and Evolve++) outperform those of the traditional learning paradigm with independent antecedent learning (eTS, Evolve). Furthermore, we note that the confusion-driven algorithm, Evolve++, gives better results (lower misclassification rates) compared to the global error-based algorithm, Evolve+.

In the rest of this section, we will omit in the presented results the methods Evolve-noBuffer and Evolve+. The focus will be placed on three models: eTS (as reference model), Evolve, and Evolve++.

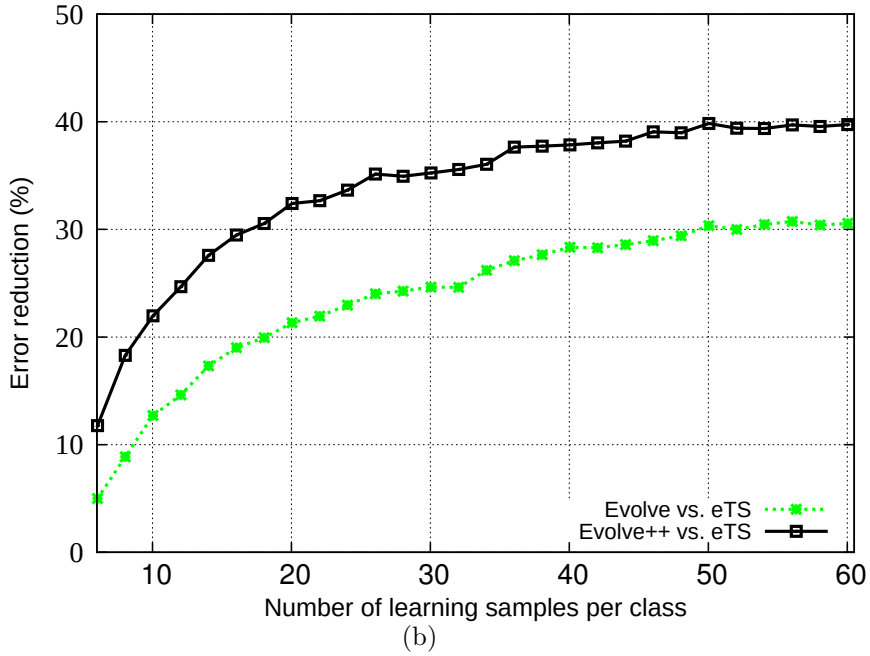
As mentioned before, our models have been tested on several benchmark datasets that are widely used in the field of machine learning. We present in Figures 4.4, 4.5, 4.6 and 4.7, the results of the three models mentioned above for three different benchmark datasets, in addition to Sign dataset (Figure 4.4 is a repetition of Figure 4.3 after omitting two curves). In each one of these figures, the evolution of the generalization misclassification rate of the three models is presented, from the very beginning of the incremental learning process (few samples per class), until a relatively advanced learning state (limited by the size of the used datasets). We show in the same figures the relative reduction of misclassification rate that is achieved using Evolve and Evolve++ compared to the reference model, eTS.

We can generally note from these results that both Evolve and Evolve++ outperform the eTS reference model. We note also the stability of the performance of these two models. Considering the average for the four problems, the rate of misclassification errors generally decreases by almost 35% by Evolve++ compared to results of eTS.

From the relative error reduction curves, we see that the difference between eTS and our model at the beginning of the incremental learning process is not as large as it becomes later in the learning process. The reason of this phenomenon is that the more sophisticated antecedent structure requires more time (i.e. more learning samples) in order to get relatively stable and thus causing less perturbation to the consequent learning. Therefore, as it can be told from the figures, the relative error reduction increases when the number of learned instances increases.

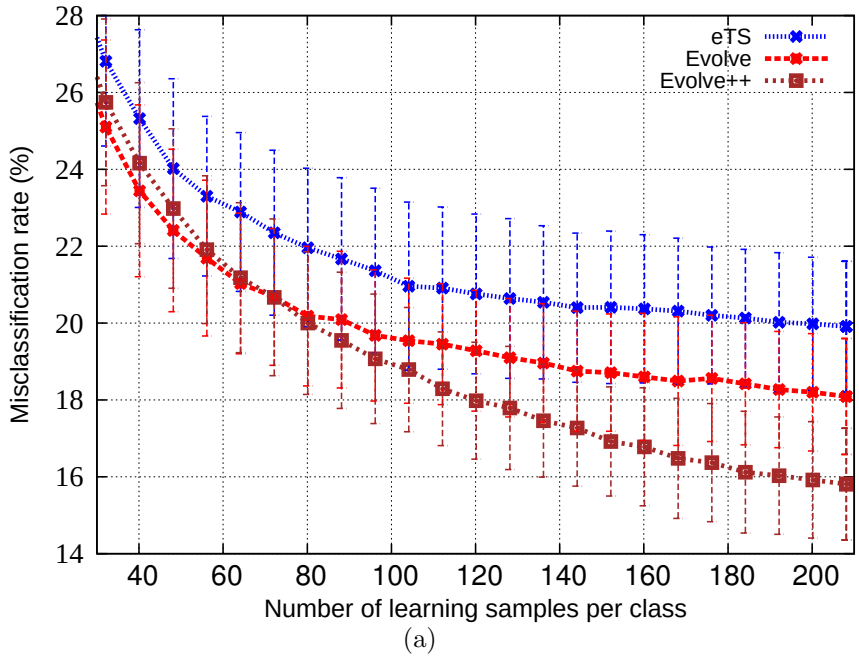



(a)

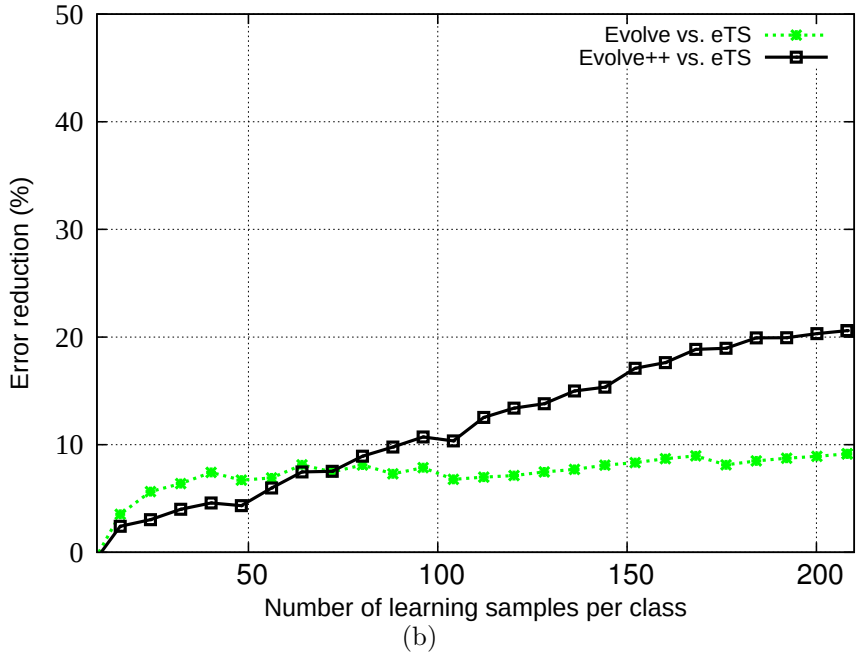



(b)

Figure 4.4: (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (Sign dataset)

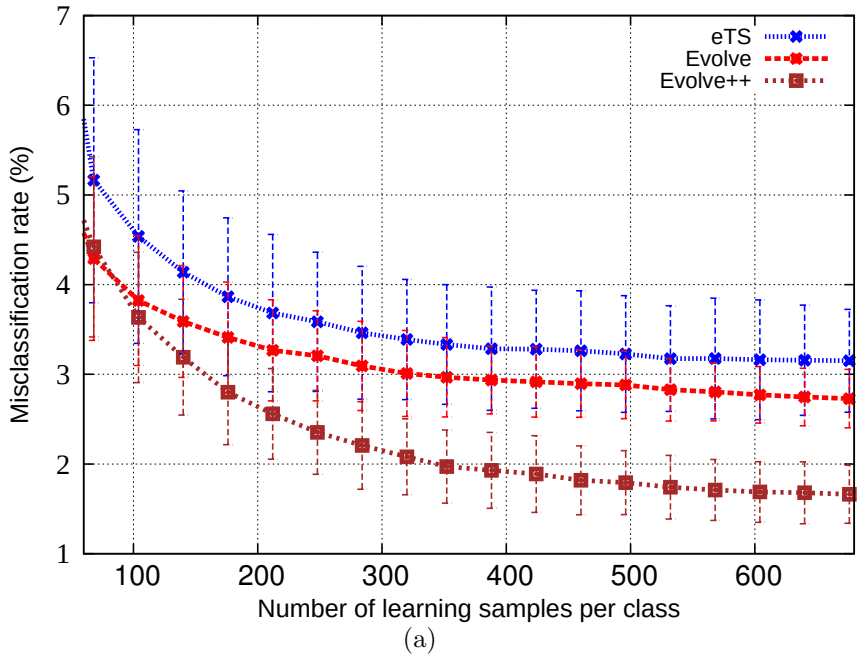



(a)

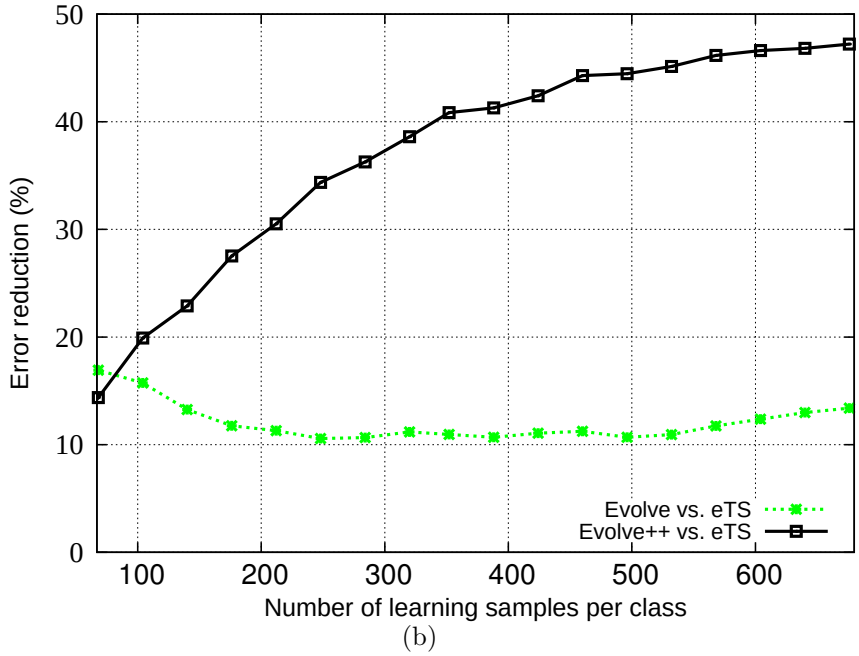



(b)

Figure 4.5: (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (CoverType dataset)

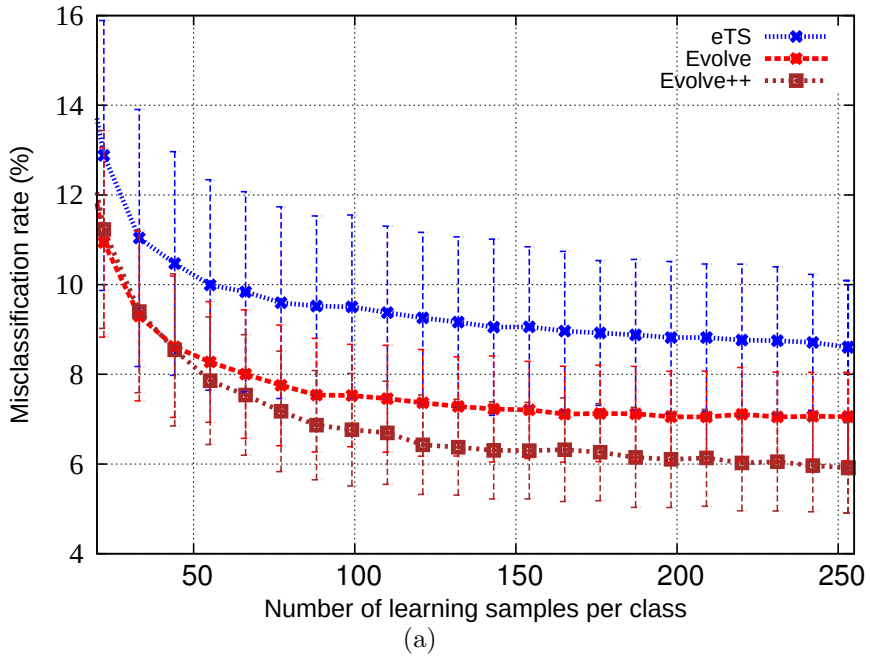



(a)

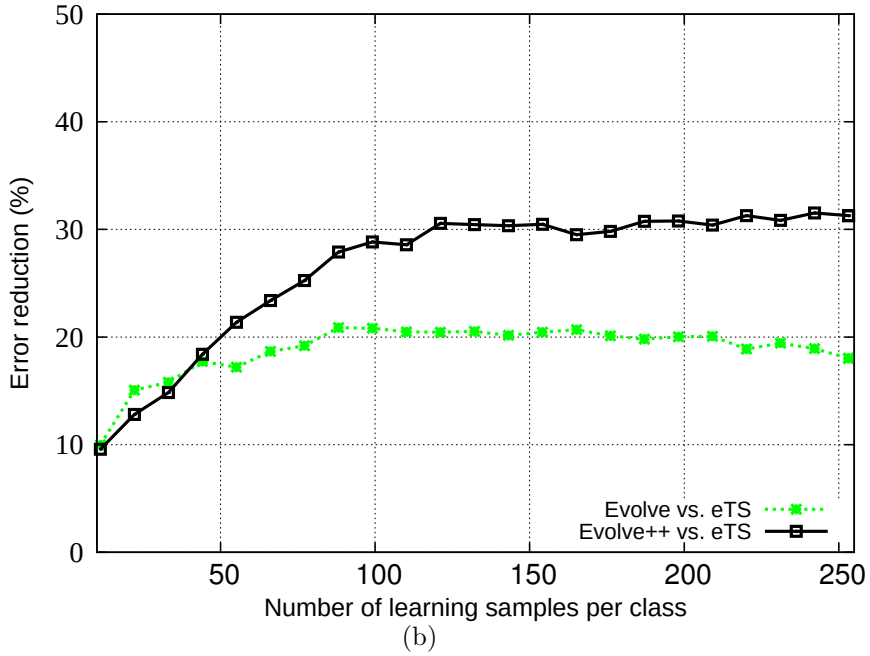



(b)

Figure 4.6: (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (PenDigits dataset)




(a)




(b)

Figure 4.7: (a) Evolution of performance during the incremental learning process and (b) Evolution of relative reduction in misclassification rates compared to the reference model (Segment dataset)

It can be noted that Evolve++ does obviously better than Evolve for the four problems thanks to the confusion-driven antecedent adaptation. As explained in the precedent chapter, avoiding useless modification on the antecedent structure results in more stability in the consequent learning, and thus a better overall precision. This difference makes Evolve++ goes generally faster and higher in the error reduction curves compared to Evolve. On the other hand, the fact of focusing on confusion classes in the antecedent adaptation of Evolve++ reduces considerably the misclassification rate for the four classification problems.

### 4.4.2 Results for different consequent structures and learnings strategies

Several variants of consequent functions and their learning algorithm can be used in our evolving models as mentioned in the precedent chapter. We have measured the impact of these variants on the classification performance for the different datasets. Table 4.3 shows a comparison between zero-order and first-order model and between local consequent optimization and global consequent optimization (see Section 3.3.3). For simplification, we only show the results related to Evolve model. A brief summary of the characteristics (number of classes, features and instances, respectively) of each dataset is given in the first line of the table.

We can note that first-order models are more performant than zero-order ones, but with the price of having more parameters. On the other hand, we note that both local and global consequents optimization give almost the same results, with the advantage of having less parameter using the local optimization. Thus, we choose in the next experiment (Table 4.4) a local consequent optimization algorithm. (Local consequent learning had been also used in the results presented in Section 4.4.1)

We sum up in Table 4.4 the misclassification rates achieved by eTS, Evolve and Evolve++ at the end of the incremental learning process in our experiments for the two types of consequences.

We present also the results of the two traditional batch classifiers (MLP and K-NN) on the same datasets. To be comparable with the incremental protocol, 75% of each dataset is used to train these classifiers and the rest is used in the evaluation. As aforesaid, the algorithms provided by the open source WEKA workbench have been used for the batch tests. For the K-NN classifier, the tests have been repeated

| Consequent Type \ Dataset (#C,#F,#I) | | PenDigits (10,16,10992) | CoverType (7,54,2100) | Segment (7,19,2310) | Letters (26,16,20000) | Sign (25,10,2500) |
|---|---|---|---|---|---|---|
| 0-Order | Global Learning | 4.04 ± 0.41 | 19.67 ± 1.59 | 7.90 ± 1.18 | 12.05± 0.37 | 6.98 ± 0.91 |
| | Local Learning | 4.02 ± 0.47 | 20.07 ±1.60 | 7.69 ± 1.05 | 12.11 ±0.48 | 7.01 ±0.95 |
| 1-Order | Global Learning | 3.00 ± 0.36 | 18.45 ± 1.55 | 7.00 ± 1.19 | 10.86 ± 0.49 | 4.16 ± 0.63 |
| | Local Learning | 2.75 ±0.38 | 18.03 ±1.60 | 7.17 ± 1.13 | 11.05 ± 0.49 | 4.18 ±0.72 |

Table 4.3: Misclassification rates for different consequent types (Evolve algorithm)

for different values of k and the best results are selected. One hidden layer had been used in the MLP neural network. The number of neurons of the hidden layer had been optimized in each experiment.

We note from table 4.4 that the superiority of our two algorithms, and especially Evolve++, is even more considerable for zero-order models. The enhanced antecedent structure and then the confusion-driven antecedent learning allows Takagi-Sugeno models to achieve high precision even with simple consequent structure.

For example, for Letters dataset, we notice that the misclassification rate achieved by zero-order version of eTS , 36.64%, is reduced to 10.05% using Evlove++ model with zero-order consequences.

Considering the results of MLP and K-NN classifiers, we note that the performance obtained by our first-order evolving TS models is completely comparable to that of some well-known batch classifiers.

The same results summarized in Table 4.4 are illustrated as charts in Figures 4.8, 4.9, 4.10, 4.11 and 4.12.

| Classifier \ Dataset (#C,#F,#I) | | PenDigits (10,16,10992) | CoverType (7,54,2100) | Segment (7,19,2310) | Letters (26,16,20000) | Sign (25,10,2500) |
|---|---|---|---|---|---|---|
| K-NN | | 2.3 | 18.5 | 8.0 | 20.9 | 4.6 |
| MLP | | 4.5 | 17.4 | 4.6 | 10.2 | 7.4 |
| eTS | 0-order | 20.59 ± 0.75 | 28.77 ± 1.82 | 21.95±1.71 | 36.64 ± 0.81 | 15.23 ±1.20 |
| | 1-order | 3.05 ± 0.32 | 19.96 ±1.38 | 8.53 ± 1.42 | 20.22 ±0.57 | 6.05 ±0.88 |
| Evolve | 0-order | 4.02 ± 0.47 | 20.07 ±1.60 | 7.69 ± 1.05 | 12.11 ±0.48 | 7.01 ±0.95 |
| | 1-order | 2.75 ±0.38 | 18.03 ±1.60 | 7.17 ± 1.13 | 11.05 ± 0.49 | 4.18 ±0.72 |
| Evolve++ | 0-order | 1.88 ± 0.33 | 16.92 ±1.70 | 6.19 ± 0.88 | 10.05 ±0.35 | 5.10 ±0.79 |
| | 1-order | 1.64 ± 0.31 | 15.81 ±1.91 | 5.95 ± 0.95 | 9.52 ± 0.31 | 3.63 ±0.59 |

Table 4.4: Misclassification rates for batch and incremental classification systems

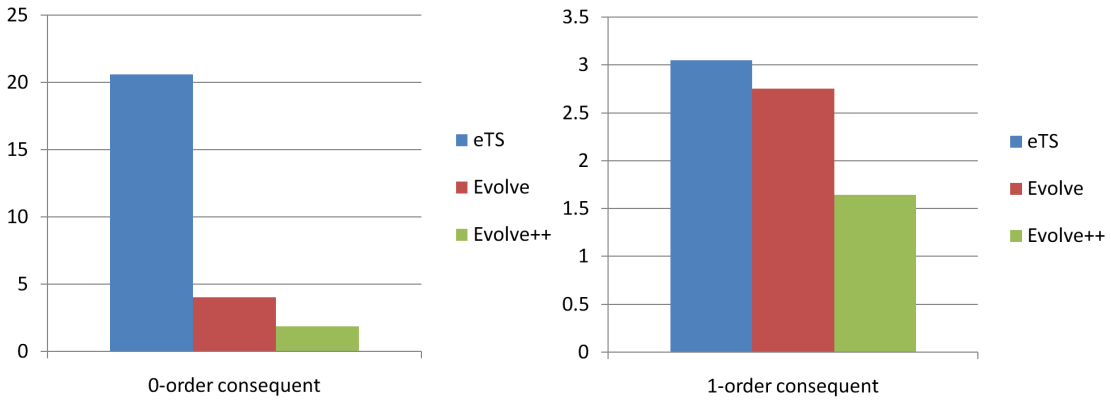



Figure 4.8: Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (PenDigits dataset)

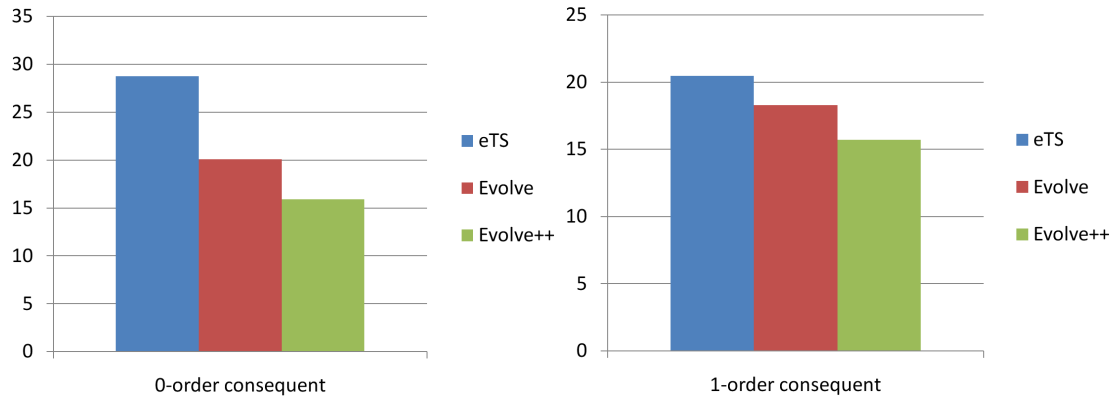



Figure 4.9: Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (CoverType dataset)

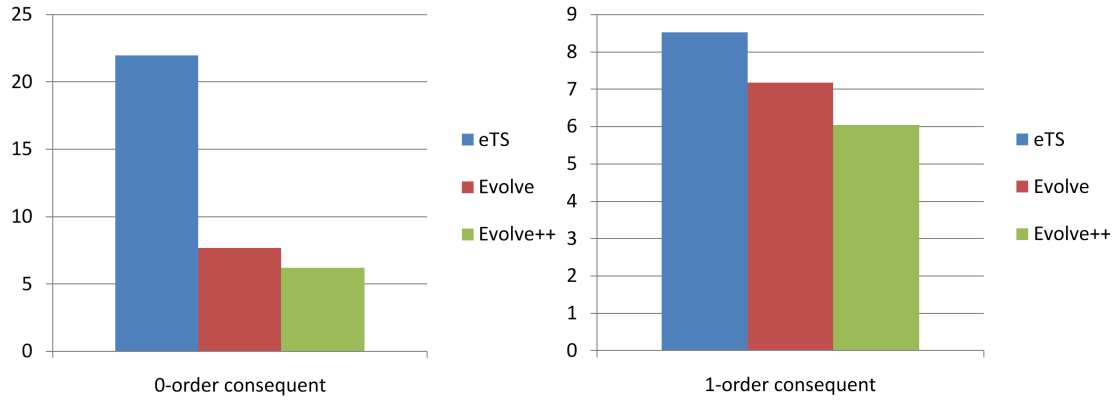



Figure 4.10: Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Segment dataset)

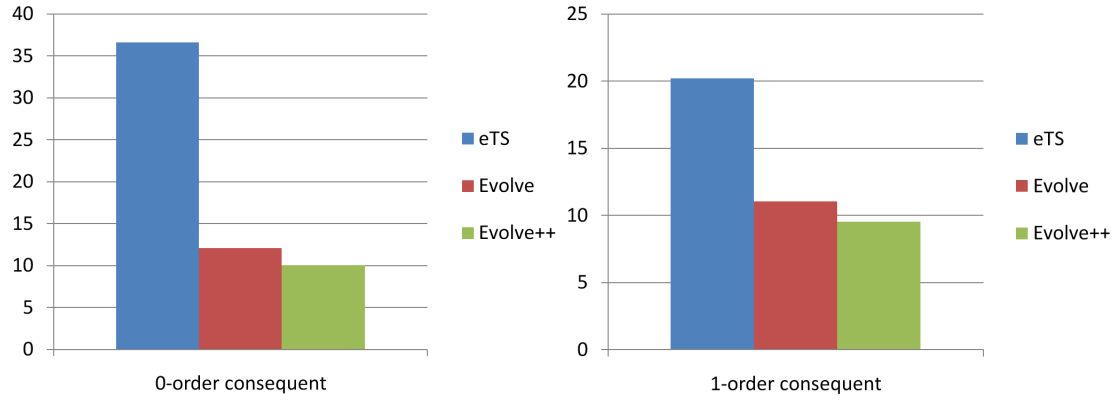



Figure 4.11: Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Letters dataset)

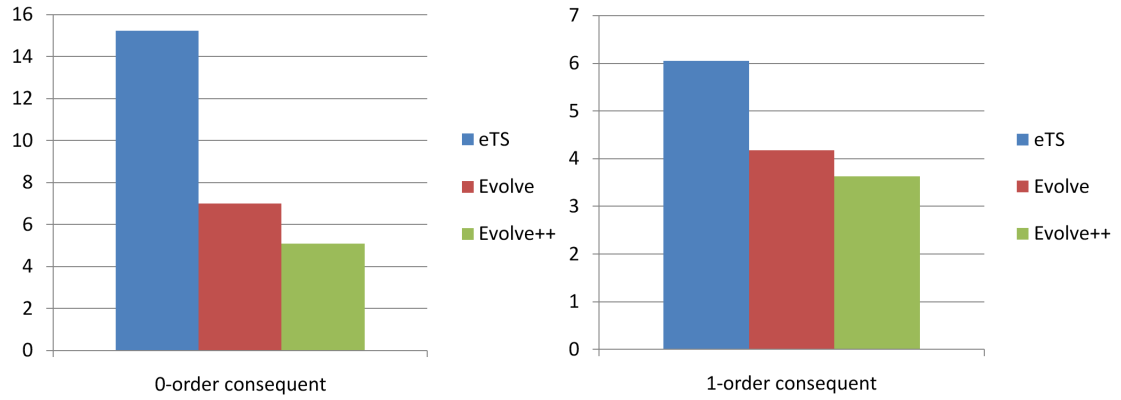



Figure 4.12: Misclassification rates of eTS, Evolve and Evolve++ models at the end of the incremental learning process (Sign dataset)

### 4.4.3 Correction efforts required by the different algorithms

All the above-mentioned recognition rates have been calculated using a separated test dataset to measure the evolution of performance of the classification systems during the incremental learning process, and more importantly, the difference between these generalization rates after a significant period of learning (summarized in Table 4.4). Obviously, the first learn/test protocol (Figure 4.2 (a)) presented in Section 4.3 has been used in the presented results.

While the incremental learning process is supposed to be supervised, a validation mechanism must be coupled to the system in order to correct the potential misclassification errors during the learning. This validation mechanism can be implemented in different manners depending on the application. In some applicative contexts, the correction action is manually performed by the user when observing a misclassification error. Therefore, it is useful to measure the number of misclassification errors committed on the incoming learning instances, and must be corrected by the validation mechanisms. This information shows the number of correction interventions required by each evolving classifier in order to achieve the misclassification rates presented in Table 4.4. The second learn/test protocol (Figure 4.2 (b)) presented in Section 4.3 is designed to provide this statistic. As mentioned in Section 4.3, the second protocol is essentially adopted when dealing with a small dataset that is entirely used for the learning process, so no test dataset is available. In this case, this protocol can give a rough idea about the generalization performance of the classifier.

In our experiments, we have used the first protocol in order to get a precise

evaluation using a relatively large test dataset. However, the number of errors committed on the learning instances can be estimated even when using the first protocol, which makes it more general than the second one.

Using the first protocol (Figure 4.2(a)), and by applying the misclassification rate estimated on the test dataset $S_{test}$ at the moment $test_i$ on the next learning subset $S_{i+1}$, we can estimated the number of recognition errors on this learning subset:

$$NbErr(S_{i+1}) = Rate(test_i) * |S_{i+1}| \tag{4.1}$$

It is important to mention that we neglect here the evolution of the classifier during the learning of $S_{i+1}$. The estimated number of errors represents then the upper bound of errors, since the learning of $S_{i+1}$ improves generally the performance as it can be noted from $test_{i+1}$.

Generally speaking, the accumulated number of learning errors between the learning samples $x_{t_1}$ and $x_{t_2}$ equals the area under the evaluation curve between $test_{t_1}$ and $test_{t_2}$. The upper bound of this area can be estimated as follows:

$$AccNbErr(t_1, t_2) = \sum_{i=t_1}^{t_2-1} Rate(test_i) * (N_{i+1} - N_i) \tag{4.2}$$

where $N_i$ represents the accumulated number of learning samples at the moment $i$.

Using this equation, we show in Figures 4.13, 4.14, 4.15 and 4.16 the accumulated number of errors and thus the correction interactions during the incremental learning process for the four classification problems.

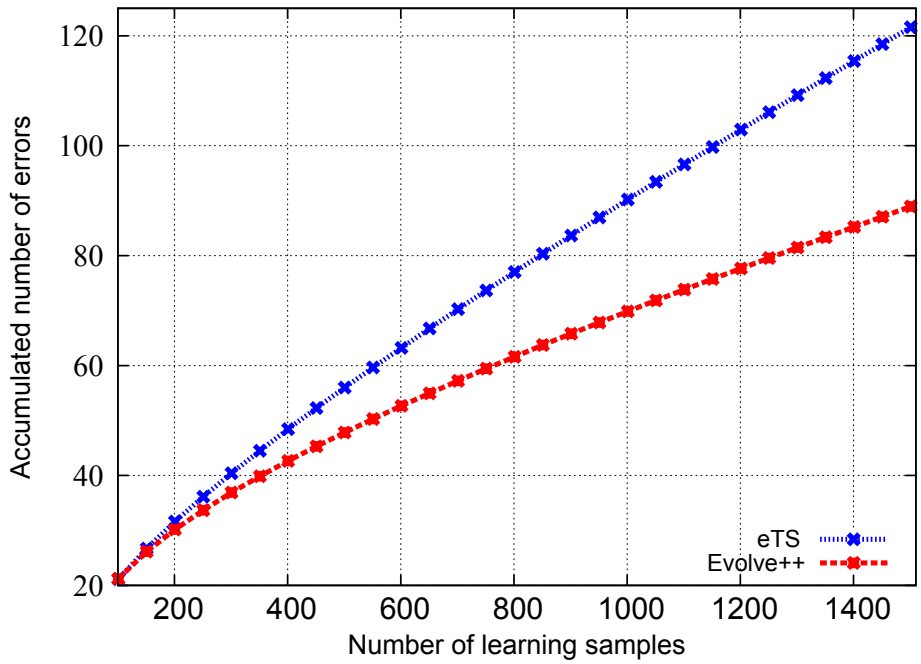



Figure 4.13: Accumulated number of misclassification errors committed on the learning samples (SIGN)

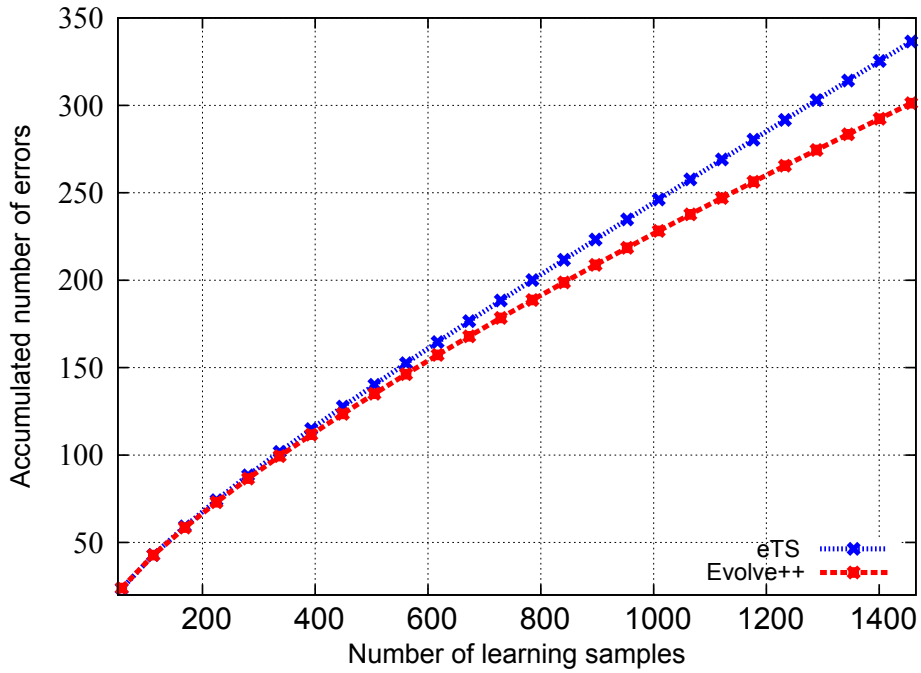



Figure 4.14: Accumulated number of misclassification errors committed on the learning samples (CoverType)

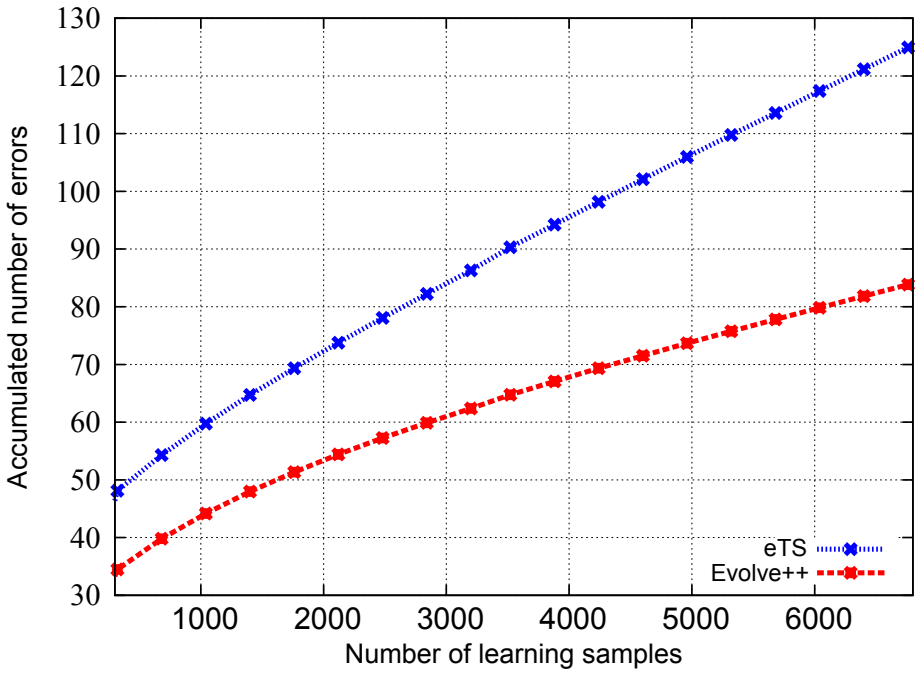



Figure 4.15: Accumulated number of misclassification errors committed on the learning samples (PenDigits)

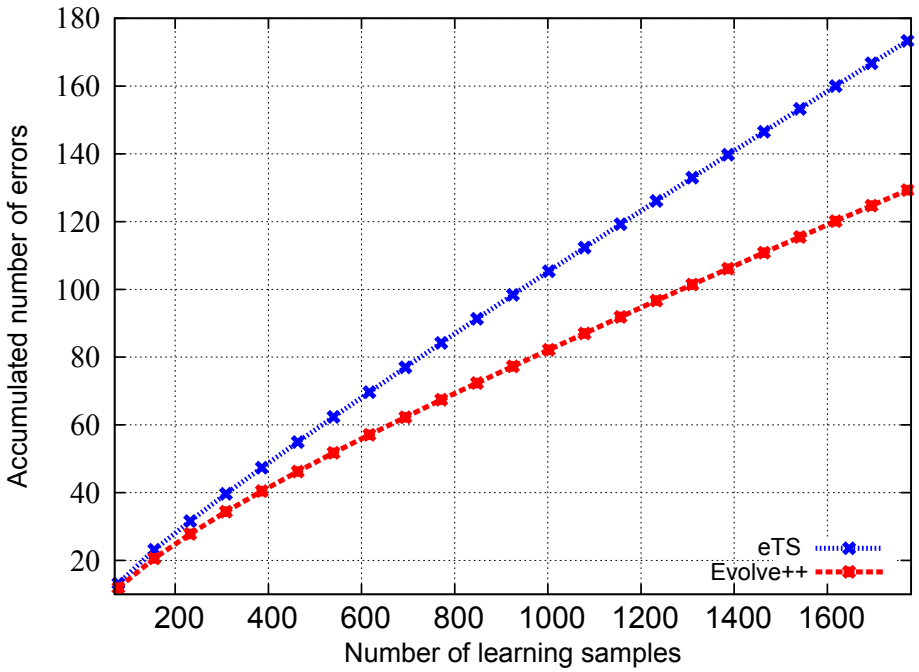



Figure 4.16: Accumulated number of misclassification errors committed on the learning samples (Segment)

For example, the number of required corrections during the incremental learning of Sign dataset is reduced from 122 using eTS to 88 using Evolve++, where the total number of learning samples is 1500. Hence, in addition to the better performance obtained at the end of the incremental learning process as shown in Table 4.4, the correction efforts required by the user during the learning is reduced. We can notice the same reduction for the other dataset.

The same results can be presented with different viewpoint if we show the performance of the classifier according to the number of required correction (the number of learning errors).

Figures 4.17, 4.18, 4.19 and 4.20 demonstrate the evolution of the misclassification rate - always obtained on an external test dataset $S_{test}$ - according to the number of corrections. We note, for example, that for Sign dataset, a misclassification rate of 7% requires about 32 corrections when using Evolve++, whereas 70 corrections are needed to achieve this rate by eTS.

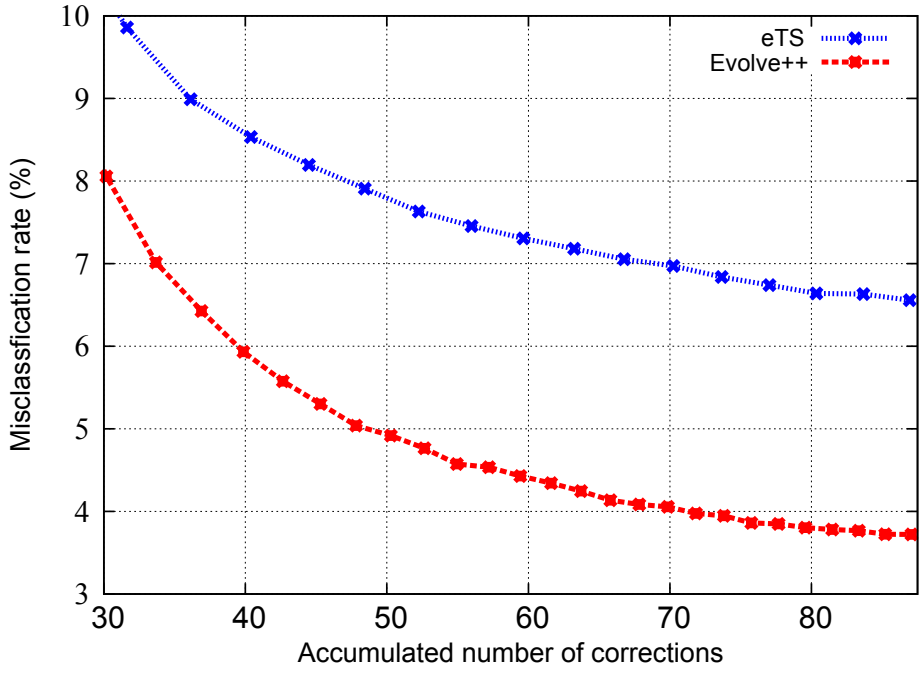



Figure 4.17: Evolution of misclassification rate according to accumulated number of corrections (SIGN)

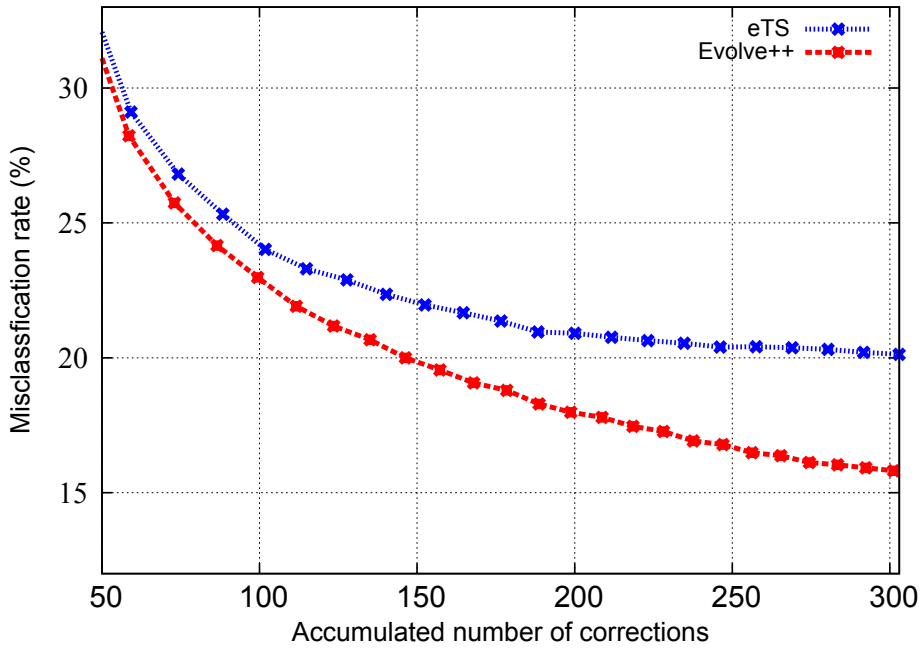



Figure 4.18: Evolution of misclassification rate according to accumulated number of corrections (CoverType)

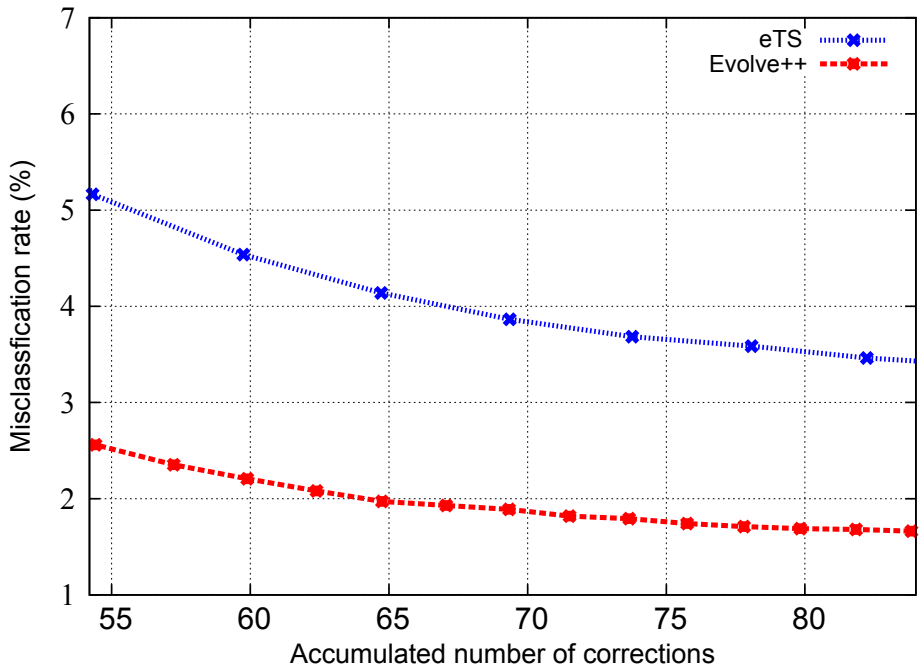



Figure 4.19: Evolution of misclassification rate according to accumulated number of corrections (PenDigits)

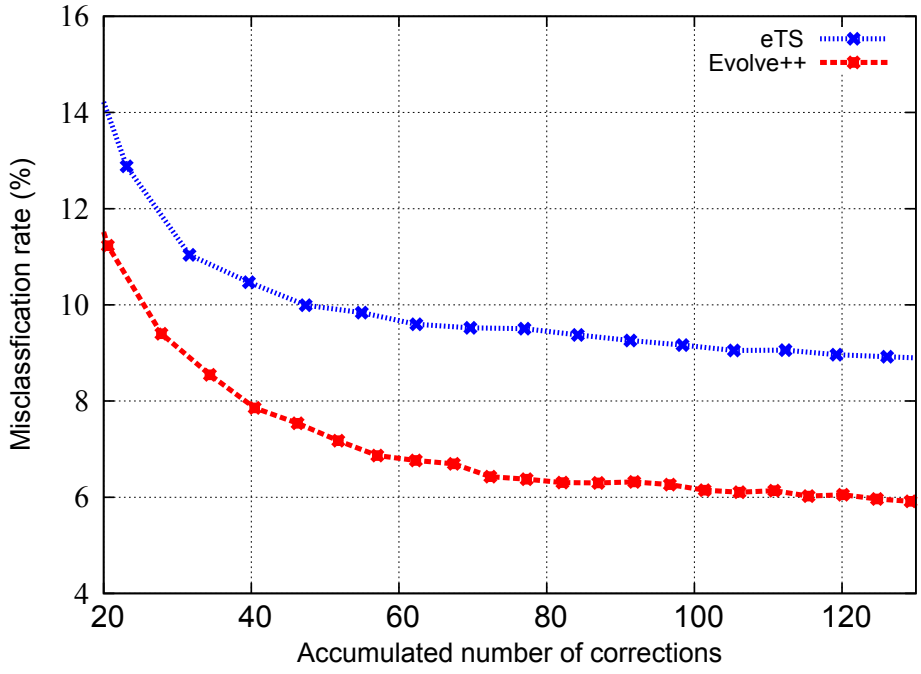



Figure 4.20: Evolution of misclassification rate according to accumulated number of corrections (Segment)

### 4.4.4 Performance evaluation for unsynchronized incoming classes

The experiments presented so far represents a simple and straightforward incremental learning scenario, in which all the classes are presented to the classifier together from the beginning of the incremental learning process. As explained before, an equitable number of samples from each class are progressively introduced to the system. The classification efficiency is repeatedly measured to evaluate the learning progress of the classifier. Evidently, a quite continuous decreasing of the misclassification error rate is expected during the incremental learning process using this scenario. The difference in the efficiency of several approached is related to the rapidity of decreasing in the misclassification rate curves and the final obtained rate (it is important to remind that the learning process is lifelong and the world "final" means here the end of the available learning dataset). We have chosen this simple scenario in the precedent experiments in order to facilitate any comparison with our algorithms by future concurrent approaches.

In addition to the continuous refinement of its knowledge base, one of the main features of an evolving classification system is the capacity of learning new un-

seen classes without suffering from the "catastrophic forgetting" phenomenon. This feature might be mandatory in several real application areas. If we consider, for example, the case of using a classification system to recognize a customizable set of handwritten shortcuts, although the incremental learning process may begin by asking the user to choose his own set of gestures, it is important to enable him to add new shortcuts at any moment.

Therefore, we present an additional experiment that imitates this real context so that the learning starts with a subset of classes, and the other classes are progressively introduced. We aim at studying the behavior of the different evolving systems and their ability to learn new class of data without fully destroying the knowledge learned from old data.

The experiment is carried out on the dataset Sign. We divide the classes in Sign into three subsets: A, B and C with 10, 7 and 8 classes, respectively. The incremental learning process starts by introducing learning samples from the classes of A. The evaluation is done during this phase using a subset from that test dataset; this subset only contains samples from the classes of A.

Then, we add in two times the classes of B followed by the classes of C. The evaluation dataset is extended at each time by adding test samples from the new learned classes.

Figure 4.21 shows the results of the experiment (It is useful to remind that these results represent the average of the results of 11 different writers, as mentioned in Section 4.2.1).

We note that our two algorithms (Evolve and Evolve++) resist better than eTS when introducing new classes thanks to the enhanced antecedent structure. Moreover, we note that Evolve++ shows faster performance recovery after introducing new classes, compared to Evolve algorithm. Concentrating the antecedent learning on the confused samples in Evolve++ results in faster fall in the misclassification curve, as it can be noted from Figure 4.21. When introducing new classes, the sophisticated forms of the new added prototypes will suffer considerable modifications for a while. Although these modifications are important in order to get the best data covering, they produce high instability in consequent learning as explained before. Here, the advantage of Evolve++ is that these significant antecedent modifications go slower for the new classes that are not confused by other classes and can be considered as "easy-to-learn" classes.

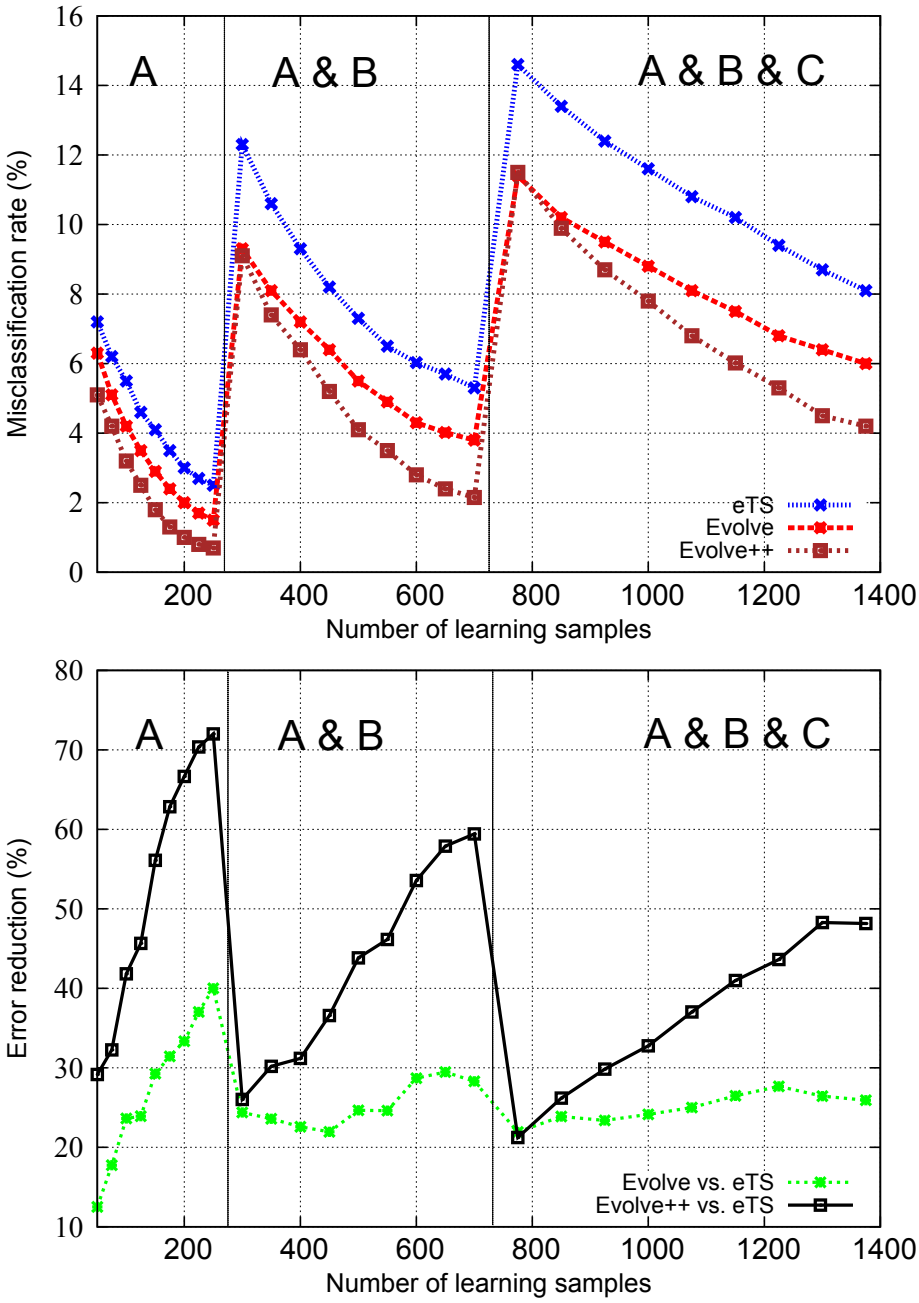



Figure 4.21: Performance stability and recovery when introducing new unseen classes (SIGN)

## 4.5 Real applications

It is important to mention that in some real applicative contexts where the human is engaged in the learning/operation process like in evolving handwritten recognition systems, in addition to the continuous adaptation of the classification system, some natural adaptation can be observed in the user behavior. For example, he may slightly change his manner of sketching some handwritten gestures to avoid misclassification errors committed so far. This phenomenon does not appear in our experiments because the collection of Sign dataset had been performed in passive mode, so that the writer does not have any feedback during the data collection session. Therefore, we can expect that the number of misclassification errors goes down in real contexts faster than the presented experimental results thanks to the natural user adaptation. A high recognition rate can be reached in handwriting recognition systems after a significant period of operation thanks to the continuous classifier learning combined with the potential user adaptation.

An experimental study in real contexts is needed in order to evaluate, on the one hand, the effect of continuous classifier learning on the user behavior, and, on the other hand, the effect of natural user adaptation on the classifier performance. We have achieved a first step in this direction by integrating our evolving classification system in different real applications related to handwriting recognition.

The first application deals with the pen-based interactive design of architectural plans. The application can interpret the handwritten sketches to walls, doors or windows based on some grammatical inference techniques. In addition, different handwritten gestures can be used to add some specific symbols to the plan, like furniture (tables, douches, sinks, etc), or external objects as trees, pools, etc. Our evolving handwritten gesture classifier is used in this application to recognize these gestures. The classifier is completely customizable; it allows each user to choose his group of gestures and assign each one to a specific symbol. He can, at any moment, add to the system a new symbol and define the corresponding handwritten gesture.

The validation/correction mechanism is implemented in this application as follows: after the recognition of a given handwritten gesture and interpreting it to the corresponding symbol which is displayed on the plan, the classifier will wait for either confirmation or correction signal. The confirmation signal is automatically sent if no “cancellation” action is performed by the user. An assumption of correct recog-

nition of the precedent received gesture will be adopted by the classifier at the next recognition request if no cancellation action has been observed. If the user rejects the classifier answer and the recognized symbol, a user-friendly interface appears to enable him to correct the error and give the true label. Even with the automatic confirmation concept, some efforts are still needed from the user to correct the erroneous responses; however, the number of errors decreases rapidly using our evolving system as we have seen in the results. The first user feedbacks had been very positive in term of the perceived classification performance and the facility of learning new gesture from very few samples. A demonstration video of this application is available at the following address: www.irisa.fr/imadoc/videos/evolve.html

Given that conducting user-centered experiments is a sensitive and sophisticated task and requires special expertise, we have created a collaboration with the research group CRPCC from LOUSTIC laboratory had taken place as a part of an ANR project called "mobisketch" (mobisketch.irisa.fr). The CRPCC group is specialized in experimental psychology and ergonomics of conception of human-machine interfaces. Thus, one of the perspectives of this work is an experimental study in real context of our evolving classifier and the user behavior and acceptability in collaboration with CRPCC research group.

We have integrated our evolving classifier in a second application that aims at mapping technical paper documents, like architectural floor plans, to numerical ones. To avoid the a posteriori verification phase and error accumulation during the analysis, the application is based on an interactive method of analysis of offline structured document where the decision process solicits the user if necessary. When an offline handwritten image of symbols is rejected by our evolving classification system, a user-intervention is raised to either give the true label of this symbol or to declare it as new class of symbols. Many correction actions can be performs and new symbols can be adding during the semi-automatic analysis process, which reduces considerably the errors in the final converted numeric document. More details about the role of our evolving classifier in this specific applicative context can be found in [60].

A third collaboration project is in progress with the Synchromedia research group (www.synchromedia.ca). One of the main research themes of Synchromedia is the development of intelligent adaptive interfaces for multi-user distributed collaborative applications. The goal of the cooperation is incorporating our evolving handwritten

gesture classifier in an online handwriting collaborative annotation platform.

## 4.6 Conclusion

We have presented in this chapter an experimental validation of the efficiency of our evolving Takagi-Sugeno system and the different learning algorithms. The experiments had been carried out on a special dataset related to the applicative context of this thesis; handwritten gesture recognition systems. Moreover, the proposed systems had been also evaluated on several well-know benchmark datasets.

It has been proven by the experimental results that improving the antecedent structure in a Takagi-Sugeno model can lead to better performance, especially if the incremental learning algorithm addresses the problem of stability in consequent learning.

We conclude from the results that the Evolve++ algorithm represents the most efficient solution for classification problems. It offers a fast incremental learning (fast fall of misclassification rate) and reduces considerably the number of classification errors during the incremental learning process, so that less external correction efforts are required.

Furthermore, we have also tested the behavior of our system when introducing new unseen classes. We have noticed that our evolving system show more stability and faster recovery compared to a well-known evolving Takagi-Sugeno system.

In spite of the very acceptable obtained performance, the main difficulty that we face when using an evolving classifier in real applications is the lack of learning samples at the beginning of the learning process and also when new classes are added. Although our system, especially with Evolve++ algorithm, learns in a fast and efficient way, it still requires a minimum number of learning samples before reaching a high classification performance. This problem also appears when introducing new classes, where a temporary but considerable rise in the misclassification rate is generally observed, as we have seen in the presented results.

In the special case of handwriting recognition, which is the main applicative motivation of this thesis, we have addressed the problem of lack of learning data in a specific manner based on our expertise in this field. In order to increase, on the one hand, the rapidity of learning at the beginning of the incremental learning process, and, on the other hand, the recovery speed after introducing new classes,

we generate artificial learning samples by applying some realistic and significant deformations on the real sample. The synthetic samples are introduced to the classifier along with the real ones and accelerate the incremental learning process. The used handwriting generation method is explained in the next chapter, with an experimental demonstration of the effect of integrating it in our evolving handwriting classification system.

# Chapter 5

# Synthetic Data Generation for Evolving Handwriting Classifiers

## 5.1 Introduction

As we have seen in the precedent chapter, our evolving system can cope with dynamic classification problems and start the learning of a new problem from scratch with few samples per class. It continuously improves its performances using new incoming learning samples indirectly provided by the user during the system operation (the samples are labeled using a semi-automatic validation/correction mechanism of the recognition responses).

Moreover, we have seen that it is possible to add new classes to the initial set of classes at any moment during the lifelong learning process; this property represents a very practical feature and leads to classification systems with high degree of dynamicity and adaptation capacity to the final user needs.

Although the dynamic nature of evolving classifiers offers many important advantages, the learning of these systems suffers from the lack of learning data. The learning process is done directly by the final user in online and interactive manner, so the quantity of learning samples is limited because it is impractical to ask the user to enter a large number of samples before obtaining a functional classifier. Therefore, the main challenge in the conception of incremental learning algorithms of evolving classification systems consists in reaching high recognition performance as fast as possible; i.e. after minimum number of learning samples.

Besides the beginning of the incremental learning from scratch, the problem of lack of data samples appears again during the learning process when new classes are added to the classifier. The evolving system is supposed to be able to learn these new classes without forgetting the old ones. However, it is difficult to completely avoid perturbations on the global performance of the classifier when adding new classes and the efforts must be focused on reducing as much as possible these perturbations.

The essential solution of the problem of incremental learning with few data consists in proposing efficient learning algorithms. For example, our algorithm Evolve++ deals with this problem by focusing on samples with high misclassification risk in order to reach high classification performance as fast as possible. We have seen in the experimental results that this algorithm does better than other approaches at the beginning of the learning process, when adding new classes, and after considerable period of learning.

In addition to the efforts of improving the classification systems and the learning

algorithms, the incremental learning process can be further accelerated and enhanced in a different manner by generating artificial learning data based on some knowledge related to the application domain.

For handwritten gesture recognition problems, this idea can be implemented by generating synthetic gestures from the available real ones after applying on them some deformations in a realistic and significant manner. Thus, when a new class of gestures is introduced to the system with few samples provided by the user, many artificial samples can be generated. They can accelerate the learning of the new gesture if the two following conditions are met:

- the generated samples are sufficiently deformed so that they bring additional information to the system;
- the generated samples are not over-deformed; they must still represent a human-like deformation of the original real gesture.

Generating artificial samples by applying human-like distortions on the real ones provided by the user represents somehow a kind of prediction of the future real samples. Obviously, this prediction cannot be exhaustive, but even partial prediction can be very helpful at the beginning of learning new gestures from very few real samples.

In this chapter, we present first in Section 5.2 a brief description of the existing handwriting generation techniques. Geometric deformations are usually applied on real handwritten symbols in order to generate synthetic ones. These deformations can be either based on class-dependent deformation models and require a learning phase, or on class-independent general deformation strategies without specific deformation models.

Then, we present a new handwriting generation technique that can be used in our evolving handwriting classification system. The proposed method does not rely on direct application of geometric deformation as the existing techniques. The deformation is instead applied on the motor representation space of the handwriting symbols. This motor representation is based on the well-known Sigma-Lognormal theory. More details about this handwriting modeling theory are provided in Section 5.3, and our handwriting generation technique based on this theory is presented in Section 5.4. Finally, we show in Section 5.5 an experimental evaluation of the pro-

posed handwriting generation technique and its impact on the incremental learning of evolving handwriting classification systems.

## 5.2 Related works

Many research works related to handwriting generation had been published during the last years. However, to the best of our knowledge, there is no previous work about the impact of using this technique in an incremental learning context.

In most of exiting approaches, artificial handwriting is generating by applying deformation on the x-y representation of the handwritten symbol, these techniques are generally called "geometric deformations".

The handwriting generation approaches based on geometric deformation can be divided into two categories: class-dependent and class-independent techniques. The two categories are explained below with some examples on each of which.

### 5.2.1 Class-dependent geometric deformations

In these approaches, a deformation model is assigned to each class of symbols, and then used to generate artificial instances of this symbol. A learning phase is required to build this deformation model.

The learning of the deformation model starts generally by the alignment of a set of samples of each symbol, then the distribution of the aligned vectors is modeled by extracting the eigenvectors and eigenvalues from the covariance matrices using Principal Component Analysis (PCA) technique, as in [11] and [12].

Different alignment techniques are used in the existing methods. In [11], the alignment is performed by elastic matching. A specific sample from each class of symbols is chosen as reference, then Dynamic Time Warping (DTW) is used to align all the x-y points of all the samples by one-to-one alignment of each of which with the reference. The principal axes of the distribution of the difference vectors are then extracted, they are referred in [11] as "Eigen-deformations of symbol". Then, synthetic instances can be generated by deforming the reference instance according to the first Eigen-deformations, as illustrated in Figure 5.1. More details about this elastic-matching based class-dependent deformation modeling can be found in [11], [61] and [13].

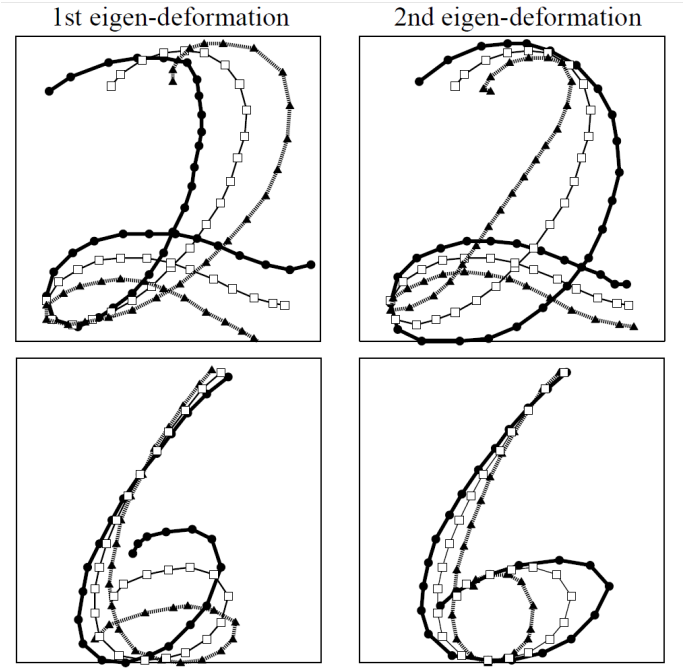



Figure 5.1: Synthetic samples (in bold) generated by deforming the reference pattern according to the first two Eigen-deformations (figure obtained from [11])

A hierarchical handwriting alignment technique is used in [12]. The handwriting trajectory is first decomposed into a sequence of pieces by landmark points, and these pieces are then matches with each other. Landmark points are mainly represented by the local extrema of curvature and the inflection points of curvature. Figure 5.2 shows some synthetic samples generated by the class-dependent deformation technique presented in [12].

The main drawback of these class-dependent deformation methods is that they require a considerable amount of samples of each class in order to create the deformation model. So they cannot be used to generate artificial instances of a new class of gestures where only few real instances are available. The two above-mentioned works are not related to incremental learning. The class-dependent deformation modeling in [11] is used to improve the performance of a static batch classifier by integrating the deformation models in the DTW-based distances. The deformation modeling in [12] is used in the context of synthesizing cursive handwriting from ASCII transcriptions. New artificial letters are generated based on the learned deformation models in order to offer some variation in the generated cursive texts. Obviously, there is no problem of lack of data in the two contexts, which makes

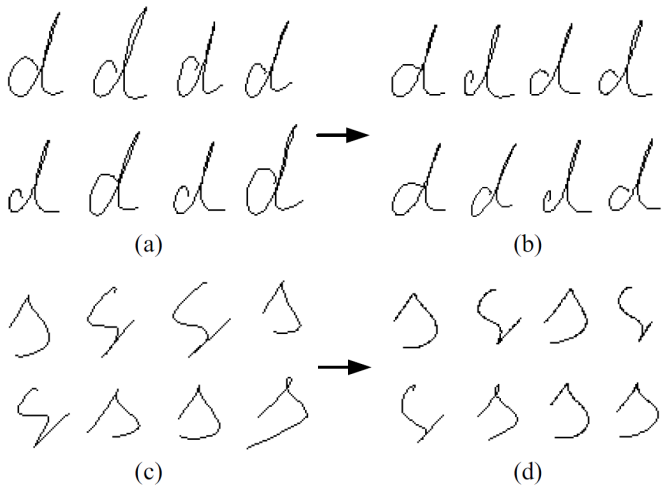



Figure 5.2: Handwriting generation using class-dependent deformation presented in [12]; (a) samples of letter "d" from training data, (b) synthesized "d" generated using trained deformation models, (c) samples of "s" from training data, and (d) synthesized "s" generated using trained deformation models


Figure 5.3: Example of class-dependent geometric deformation, the deformation observed in (a) for the letter "M" is rarely observed for the letter "H" as illustrated in (b) (figure obtained from [13])

class-dependent deformations practical for them.

Therefore, learning-based class-dependent geometric deformation cannot be used in incremental learning contexts. As mentioned above, the role of the data generation is more important at the beginning of the learning of new gestures where only few data are provided, so the deformation technique must not require a class-dependent learning phase.

Evidently, a deformation model learned using instances from a specific handwriting form cannot be applied to another one (see Figure 5.3).

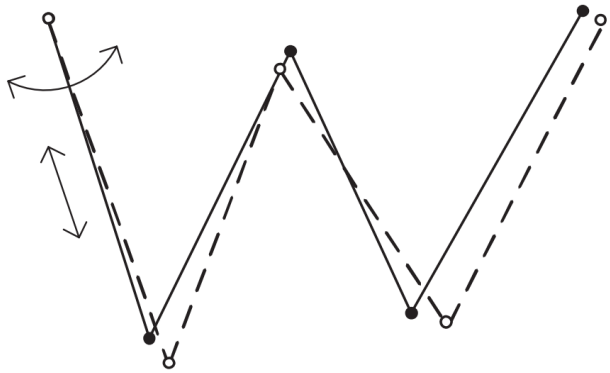


Figure 5.4: Example of class-independent geometric deformation used [14]. Small random scaling and rotation of each stroke piece are applied to each stroke.

### 5.2.2 Class-independent geometric deformations

As we have seen in Section 5.2.1, the main drawback of class-dependent handwriting deformation is the need of a learning phase, which may not be practical in some applications, like incremental learning of evolving handwriting classifiers. The second category of the existing geometric deformation techniques is the class-independent deformations. Synthetic handwritten symbols can be generated based on class-independent deformation by applying general distortions on the real symbols. The same deformation strategies are generally applied regardless the class of the symbol.

In [14], some shape distortions are applied on the letters to simulate handwriting variation in the synthetic handwriting texts. As illustrated in Figure 5.4, the strokes of each symbol are first delimited by high curvature points, then slight random rotation and scaling are applied to them. The distortion is performed at a small scale and the deformed symbol looks slightly different from the original one.

The goal of the deformation in [14] is not the prediction of human-like deformations; it only aims at slightly adding a touch of variability on the stored handwritten models of letters. However, noticeably different cursive texts can be generated by coupling this slight deformation on the letter level with other variations in letters spacing and words spacing.

More sophisticated class-independent deformation techniques have been developed by IMADOC research group during the last years [15] [62]. The deformation strategy includes two image distortions (scaling and slanting) and two online distortions based on the specificities of the online handwriting (speed and curvature).

Each distortion depends on one or more parameters. To generate a synthetic symbol from the original one, random values of these parameters are first drawn, and then the different distortions are successively applied to obtain the synthetic symbol. We explain below in some detail this class-independent geometric deformation because it is used in the experiments presented in Section 5.5 and compared to our new motor-based deformation.

The four distortions used in the above-mentioned deformation approach are as follows:

- **Uncorrelated x-y scaling:** the x-y coordinates of the points of the handwritten symbol are scaled based on two parameters $\alpha_x$ and $\alpha_y$. $\alpha_x$ represents the coefficient of scaling on the x axis, whereas $\alpha_y$ represents the coefficient of scaling on the y axis. The coordinates of the points are modified according to the following equations:

$$x'_{(t)} = \alpha_x \, x_{(t)}, \tag{5.1}$$
$$y'_{(t)} = \alpha_y \, y_{(t)}. \tag{5.2}$$

  Obviously, $\alpha_x$ and $\alpha_y$ should have different values; only simple and useless zooming effects will be obtained otherwise.

- **Slant modification:** this distortion allows generating inclined handwriting. It depends on a parameter $\alpha_s$ which represents the tangent of the slant. The y coordinates are not modified but the variation of $x_{(t)}$ depends on the value of $y_{(t)}$, as follows:

$$x'_{(t)} = x_{(t)} + \alpha_s * y_{(t)}, \tag{5.3}$$
$$y'_{(t)} = y_{(t)}. \tag{5.4}$$

- **Speed variation:** the aim of this distortion is to modify the size of vertical and horizontal parts of the symbol depending of a random parameter $\alpha_v$. These straight parts of the writing can vary without changing the global shape of the symbol. The magnitude of the speed vector $\vec{V}(t-1) = (x(t) - x(t-1), y(t) - y(t-1))$ is modified at each point depending on its

direction. If this vector is close to one of the axes then it is increased or decreased by the ratio $\alpha_v$. The coordinates of the points of the synthetic symbol are defined as follows:

$$x'_{(t)} = x'_{(t-1)} + \beta * (x_{(t)} - x_{(t-1)}), \quad (5.5)$$

$$y'_{(t)} = y'_{(t-1)} + \beta * (y_{(t)} - y_{(t-1)}) \quad (5.6)$$

$$\beta = \begin{cases} 1 & \text{if} \arg(\vec{V}_{(t-1)}) \left[\frac{\pi}{2}\right] \in [\frac{\pi}{8}, \frac{3\pi}{8}] \\ \alpha_v & \text{otherwise} \end{cases}$$

- **Curvature modification:** This distortion modifies the curvature of the strokes of the handwritten symbol. The curvature modification uses a random parameter $\alpha_c$. The curvature at the point $p(t-1)$ is defined by the angle $\hat{\theta}(t-1) = (p(t-2), \widehat{p(t-1)}, p(t))$ in $]-\pi, \pi]$. The following equation gives the position of the point $p'(t)$ depending on the two previous points and on the original curvature $\hat{\theta}(t-1)$ modified by $\alpha_c$:

$$(p'(t-2), \widehat{p'(t-1)}, p'(t)) = \hat{\theta}(t-1) - \alpha_c * 4 * \frac{\left|\hat{\theta}(t-1)\right|}{\pi} * (1 - \frac{\left|\hat{\theta}(t-1)\right|}{\pi}) \quad (5.7)$$

The four aforementioned distortion techniques are complementary; i.e. they do not generate the same deformations. They are applied in succession on the real handwritten symbol to generate synthetic ones. The distortion parameters ($\alpha_x$, $\alpha_y$, $\alpha_s$, $\alpha_v$ et $\alpha_c$) are drawn randomly in the range of possible values. These ranges are experimentally determined in order not to generate over- or under-deformed symbols. Figure 5.5 shows examples of handwritten letters generated by each distortion.

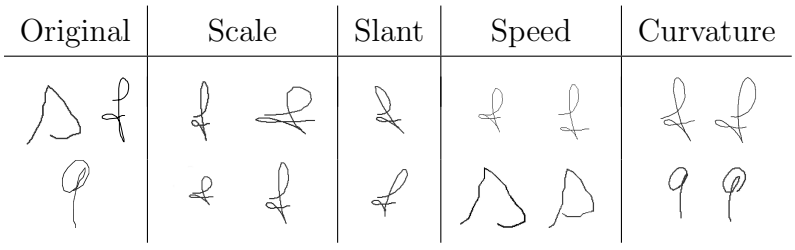



Figure 5.5: Examples of synthetic characters generated by the four distortion techniques proposed in [15]

The latter data generation approach had been used in the adaptation of multi-writer handwritten character recognition systems for specific writing style of final

users. Synthetic samples are generated from the samples provided by the final writer to adjust the classifier. Faster adaptation had been observed thanks to the synthetic samples. However, the technique had been particularly designed for the handwriting characters, and had not been used for other handwritten forms.

Setting the variation ranges of the distortion parameters ($\alpha_x$, $\alpha_y$, $\alpha_s$, $\alpha_v$ et $\alpha_c$) is very difficult if the deformations have to be applied on unknown type of handwritten gestures. Using the ranges optimized for handwritten characters on a completely different type of gestures had led to unrealistic and over-deformed shapes. The main reason of this weakness is that the four distortions work in the x-y representation space of the handwritten symbols (including the speed variation distortion which is directly induced from the x-y coordinates and does not rely on a specific writing speed model). Many studies have shown that handwriting control is a vectorial process rooted in differential geometry. Therefore, movement modeling of handwriting is believed to be more accurate and realistic than shape modeling.

### 5.2.3 Motor-based deformations

Several models have been proposed to study the human mechanism of handwriting production [63] [64] [65] [66] [67]. Different models had suggested that the production of complex handwritten forms can be considered as a vectorial superimposition in time of different strokes. The temporal overlap of these strokes is the source of smooth feature of handwriting. It had been supposed that all these sub-movements (strokes) have a stereotypical velocity pattern that can be generalized for any rapid human movement. Different models of velocity profile of a handwritten stroke had been proposed [68] [69] [70]. One of the models that had received very large acceptance and been the subject of several theoretical proofs and experimental validations is the Kinematic Theory of rapid human movements [67] [71]. It suggests that the impulse response of neuromuscular networks (the origin of rapid movements such handwriting) can be characterized with lognormal functions. The highest level of representation in the family of lognormal models supported by the Kinematic Theory is the Sigma-Lognormal model [72]. Based on the Sigma-lognormal model, each sub-movement in a handwritten trajectory can be represented by a set of six parameters $(t_0, D, \sigma, \mu, \theta_s, \theta_e)$.

Based on the abovementioned motor modeling, the variability observed in handwriting can be interpreted (and thus generated) by, on the one hand, the intrinsic

variability of the individual (virtual) strokes, and, on the other hand, the variation occurring in the time plan of superimposition of these strokes [73]. Using the Sigma-lognormal model, the former source of variability can be simulated by applying some variation on the peripheral parameters $(\sigma, \mu)$ and the control parameters $(D, \theta_s, \theta_e)$, while the latter one can be reproduced by slightly varying the parameter $t_0$ of each stroke.

We present in this chapter an automatic handwritten gesture generation method based on the Sigma-lognormal parameters. This class-independent motor-based deformation is integrated in the incremental learning process of our evolving handwriting classification system to improve the learning of new classes from few real samples.

## 5.3 Sigma-Lognormal model

The Sigma-Lognormal [70] model consider the neuromuscular system as a cascaded network of a large number of subsystems coupled through a proportionality law. Applying the central limit theorem to this representation shows that such a system will produce elementary movement units having a lognormal speed profile. Such a unit, referred to as a neuromuscular component, is scaled and time-shifted by values labeled respectively as $D$ and $t_0$, and has a lognormal shape characterized by the time delay $(\mu)$ and the response time $(\sigma)$ of the neuromuscular system involved in its production. Moreover, for planar motions, it is hypothesized that each neuromuscular component is made along a fixed pivot and hence, has a circle-arc trajectory starting at an angle $\theta_s$ and ending at an angle $\theta_e$.

The speed profile and the angular variation of a component are described by (5.8) and (5.9) where $P_j$ is the vector of Sigma-Lognormal parameters of the $j^{th}$ component (i.e. $P_j = [t_{0j}\ D_j\ \sigma_j\ \mu_j\ \theta_{sj}\ \theta_{ej}]$).

$$v_t(t; P_j) = \frac{D_j}{\sigma_j(t - t_{0j})\sqrt{2\pi}} e^{-\frac{\left[ln(t-t_{0j})-\mu_j\right]^2}{2\sigma_j^2}} \tag{5.8}$$

$$\phi(t; P_j) = \theta_{sj} + \frac{\theta_{ej} - \theta_{sj}}{D_j} \int_0^t v_t(\tau; P_j) d\tau \tag{5.9}$$

The Sigma-Lognormal model generates complex motions by a vectorial summation of time overlapping neuromuscular components. Once a parameter set is

defined, velocity in Cartesian space can be computed using (5.10).

$$v_x(t;P) = \sum_j v_{tj}(t;P_j)cos(\phi_j(t;P_j)) \quad , \quad v_y(t;P) = \sum_j v_{tj}(t;P_j)sin(\phi_j(t;P_j)) \tag{5.10}$$

The x-y coordinates of the trajectory of the handwritten symbol can be obtained from the global velocity profile given in Equation 5.10 in a straightforward manner as follows:

$$x(t;P) = \int_0^t v_x(\tau;P)\ d\tau \tag{5.11}$$

$$y(t;P) = \int_0^t v_y(\tau;P)\ d\tau \tag{5.12}$$

To sum up, any handwritten gesture is considered as the result of the superimposition of several strokes. Each stroke is described according to the Sigma-lognormal model by six parameters $(t_0, D, \sigma, \mu, \theta_s, \theta_e)$. These parameters reflect the motor control process and the neuromuscular response. As explained in [73], each stroke is produced by a volitional Dirac-Impulse signal occurring at a time stamp $t_0$ and sent to the input of the neuromuscular network that controls the arm movement. The impulse signal embeds the geometrical features of the stroke, which are the length $D$, the starting directional angel $\theta_s$ and the ending directional angel $\theta_e$ of the stroke. The velocity lognormal profile produced by the neuromuscular system is affected by the logtime delay $\sigma$ and the logresponse time $\mu$. The latter two parameters are relatively independent from the specific performed handwriting movement, but rather related to the writer (muscle conditions, age, stress ...).

Figure 5.6 shows two examples of the reconstruction of handwriting movements using the Sigma-Lognormal modeling. We see from Figure 5.6(a) and Figure 5.6(b) that the original velocity profile had been precisely approximated by the superimposition of a set of time-shifted lognormal functions.

Handwriting modeling by the Sigma-lognormal theory had been widely accepted thanks to its neuromuscular basis and the several experimental studies that have shown high accuracies of handwriting reconstruction. However, the main next challenge had been the development of an automatic and efficient algorithm that can extract from a given handwriting trajectory the parameters of the lognormal profiles that can reproduce this trajectory.

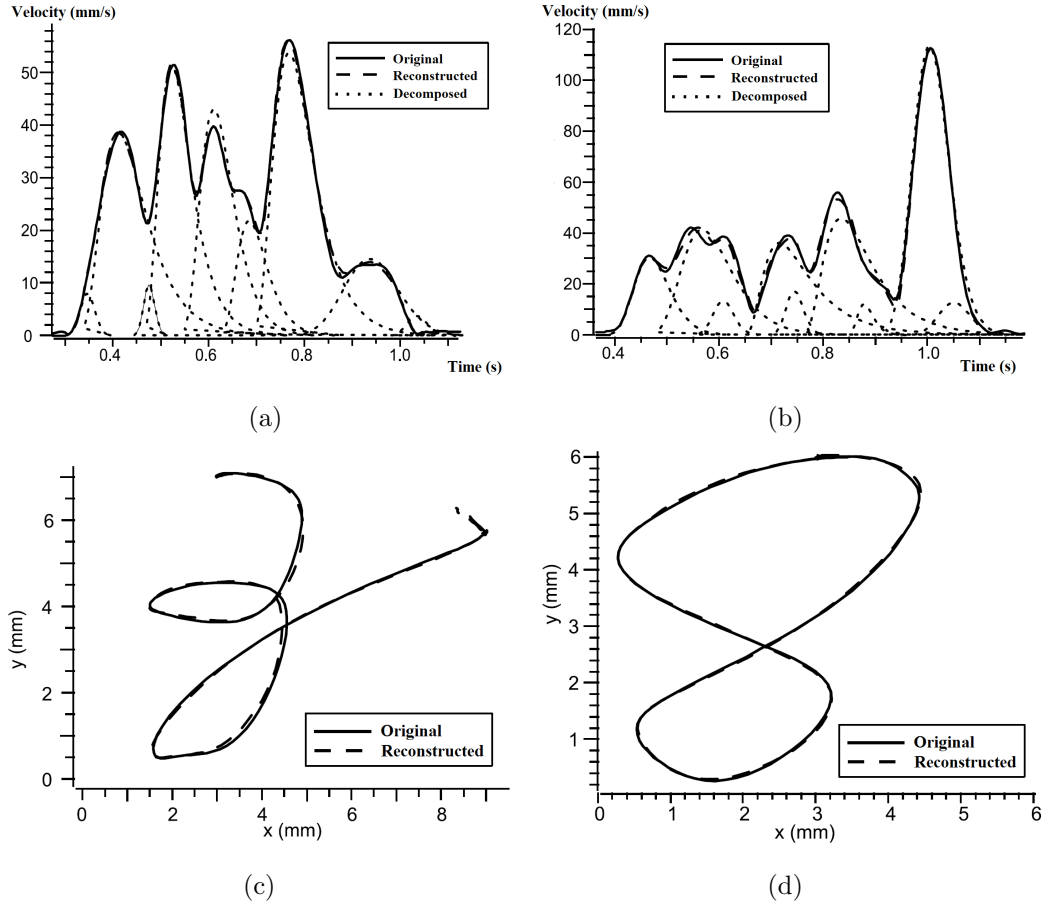



Figure 5.6: Example of the reconstruction of handwriting movements using Sigma-Lognormal model (figure obtained from [16])

A robust and automatic algorithm had been recently proposed for the abovementioned purpose [74]. It extends the principals of XZERO algorithm - that had been designed for simple strokes - to deal with complex handwriting trajectories. This automatic parameter extraction algorithm offered the possibility of using the Sigma-lognormal modeling in handwriting-related automatic systems, such as automatic handwriting generation.

Realistic and human-like deformations can be applied on a handwritten symbol by first extracting its sigma-lognormal parameters, then introducing some variations on these parameters, and finally regenerate a new symbol from the modified parameters.

Handwriting generating based on Sigma-lognormal modeling had been studied in [75]. It had been demonstrated that handwriting datasets of unlimited size can be generated from few specimens by applying variations on the Sigma-lognormal parameters as mentioned above. However, this generation technique had not been integrated in a global handwriting system and the quality of the generated data had only been visually evaluated.

In our work, we incorporate a handwriting generation technique using class-independent lognormal-based deformations in the incremental learning of evolving handwritten gesture classifiers. Thanks to the automatic parameter extraction algorithm, the data generation is performed in an automatic manner as a part of the learning process. Deformations rooted in the motor representation space of the handwritten gestures are supposed to be more realistic than geometric deformations and thus more valuable in the learning process.

In addition to the great advantage of integrating the lognormal-based handwriting generation technique in our evolving handwriting classifier, an objective and numerical evaluation of the quality of generated data is provided for the first time, to the best of our knowledge. It replaces the visual subjective evaluation used so for in several works [75] [12]. The generated handwritten samples are considered realistic as much as they help the classifier to predict future real samples from the same class of gestures. The capacity of prediction is translated by the improvement of recognition performance of the classifier, which can be measured using an independent dataset as we have seen in the precedent chapter.

## 5.4 Synthetic handwritten gesture generation using Sigma-Lognormal model

Synthetic handwritten symbols can be generated by applying some distortions on real ones. Most of the existing approaches apply these distortions on the x-y coordinates of the symbol, as we have seen in Section 5.1. Since this kind of distortion does not rely on a handwriting generative model, one can obtain unrealistic forms that do not correspond to natural human handwriting sketching.

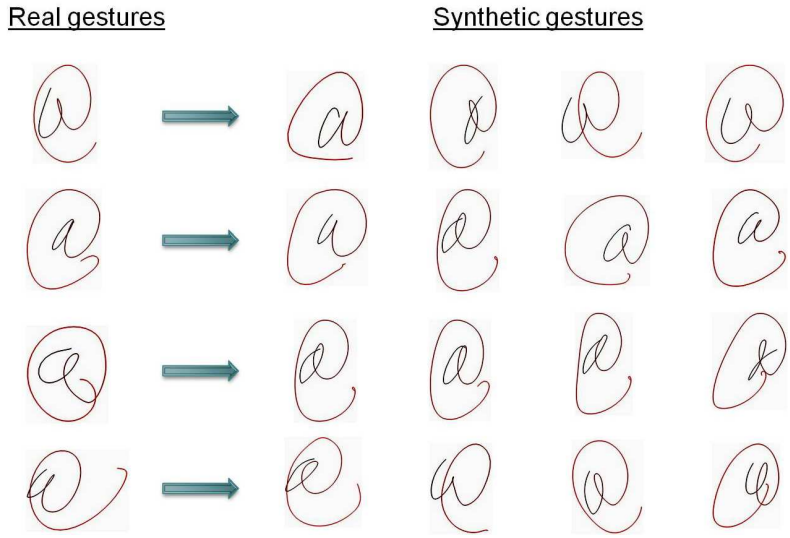
We have seen in the precedent section that any handwriting trajectory can be modeled and reconstructed by a set of lognormal functions (profiles). Each profile is defined by six parameters according to the Sigma-Lognormal theory. Thus, besides the geometrical x-y representation, each handwritten symbol can be represented in a motor representation space by a number of parameters proportional to the number of lognormal profiles used to produce the symbol. For example, a symbol produced by four lognormal profiles can be characterized and regenerated by a set of 24 parameters. A one-to-one map can be defined between the geometric x-y space and the Sigma-lognormal parameters space. The lognormal profiles of a given handwritten symbol can be automatically extracted by the extraction algorithm mentioned above, whereas the corresponding handwriting trajectory can be generated from a give set of lognormal profiles in straightforward manner using Equations 5.11 and 5.12.

Given that the transformation between the two representation spaces can be automatically performed in the two directions, an automatic synthetic handwriting generation can be defined by applying distortions on the Sigma-lognormal representation of the given real symbol instead of the traditional direct geometrical distortions. We believe that distortions obtained by applying some variations on lognormal parameters are more realistic that those obtained using direct geometrical distortions.

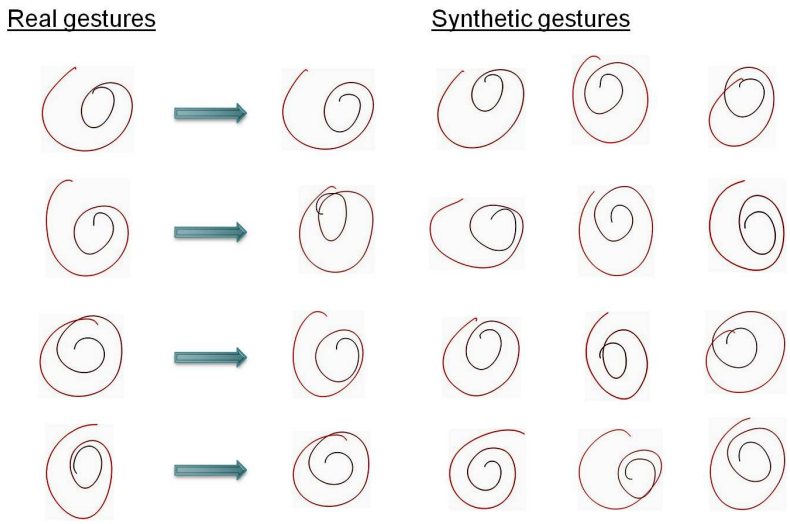
The lognormal-based generation mechanism can be simply described as follows:

- the lognormal parameters of the real handwriting gesture are first extracted;
- then, these parameters (or some of them) are randomly modified within specific variation intervals;
- a synthetic gesture is then generated from the modified lognormal parameters.

The resemblance between the synthetic and the real gestures is controlled by the variation intervals. Thus, a suitable setting of these intervals is required in order to avoid over-deformed gestures.




(a)




(b)

Figure 5.7: Some examples of generated gestures

We show in Figure 5.7 some examples of the artificial gestures that can be generated by applying modifications on the Sigma-lognormal parameters. We can note that the real gestures can be almost predicted from the synthetic ones.

## 5.5 Evolving handwriting classifier learning acceleration using synthetic data

As mentioned in Section 5.1, in the context of incremental learning of evolving systems, one can somewhat overcome the problem of lack of learning samples at the beginning of learning a new class by generating in an "intelligent" manner a number of synthetic samples. For an evolving handwriting classifier, the abovementioned Sigma-lognormal based technique for synthetic gesture generation can be incorporated into the incremental learning process. The handwriting generation is automatically performed in transparent manner with no user intervention. Figure 5.8 shows the different steps of the generation method. For each incoming sample, its Sigma-lognormal parameters are first extracted. These parameters are modified and a number of synthetic samples are generated. The original sample along with the synthetic ones are then introducing to the incremental learning algorithm.

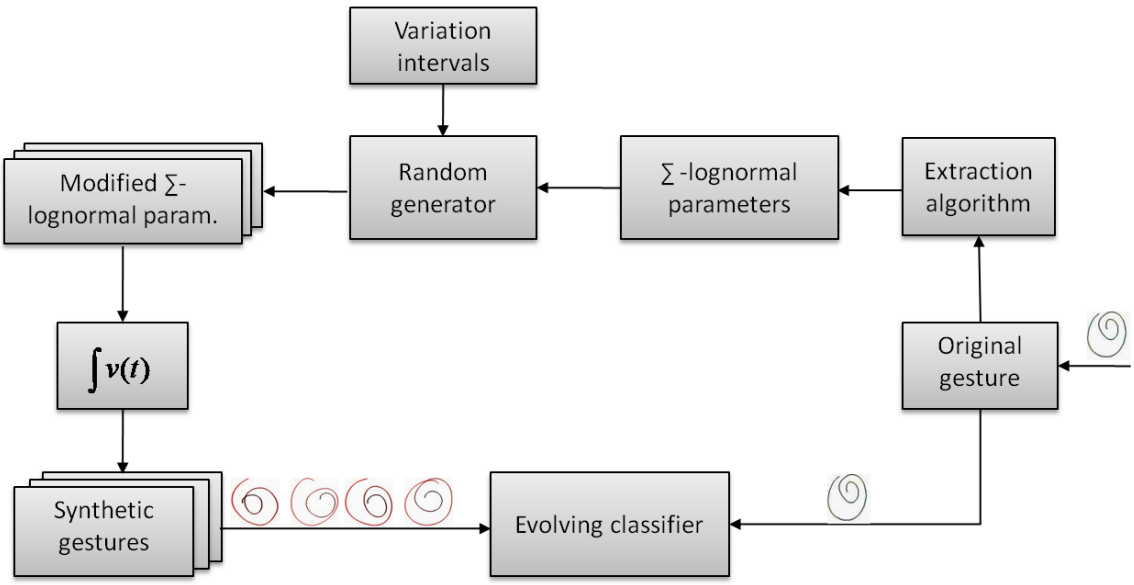



Figure 5.8: Incorporating lognormal-based data generation in the learning of evolving handwriting classifiers

Using a visualization interface developed by Scribens laboratory that allows an interactive modification of Sigma-lognormal parameters of a given handwritten gesture, we have experimentally studied the valid variation intervals of the six parameters within which the generated gesture is generally considered similar (from a human viewpoint) to the original one. The optimal variation intervals (relative or

absolute) that we have found are presented in the next section. The impact of the synthetic gestures on the classifier recognition performance is also presented in the next section.

## 5.6 Experimental results

Before presenting the experimental results, we would like to thank Scribens laboratory to offer us access to some of their algorithms and programs including: the Sigma-lognormal parameter extraction algorithm, the algorithm of handwriting trajectory generation from Sigma-lognormal profiles, and an interactive visualization software. The following experimental work could not be achieved without these indispensable components and tools.

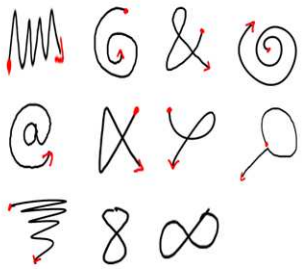


Figure 5.9: The handwritten gestures used in the handwriting generation experiments (a subset of Sign dataset)

The experiments are carried out on a subset of Sign dataset that contains the more complex gestures. It is composed of 11 classes of gesture illustrated in Figure 5.9. We have followed the same experimental protocol used in the precedent section.

We compare the lognormal-based handwriting generation method to the one based on geometric distortions explained in Section 5.2.2. Therefore, three performance curves are presented in the figures:

- Evolve++: our evolving classification approach with Evolve++ algorithm presented in the precedent chapter. Only real samples are considered (no synthetic data);

- Evolve++&Geo: synthetic samples are generated by geometric distortions (see Section 5.2.2) and used along with real ones to train Evolve++ system;
- Evolve++&Sigma: here, the synthetic samples are generated by the lognormal-based method.

For the last two approaches, 10 synthetic samples (gestures) are generated for each real learning sample.

We present the results for two different experimental scenarios: the 11 gestures are introduced together in the first scenario, while the gestures are progressively introduced in the second one. We measure in the former the impact of the synthetic samples at the beginning of the process of learning from scratch, while the latter scenario aims at showing the impact of these synthetic samples when introducing new gestures. The results of the first scenario are presented in Figure 5.10, and those of the second one in Figure 5.11.

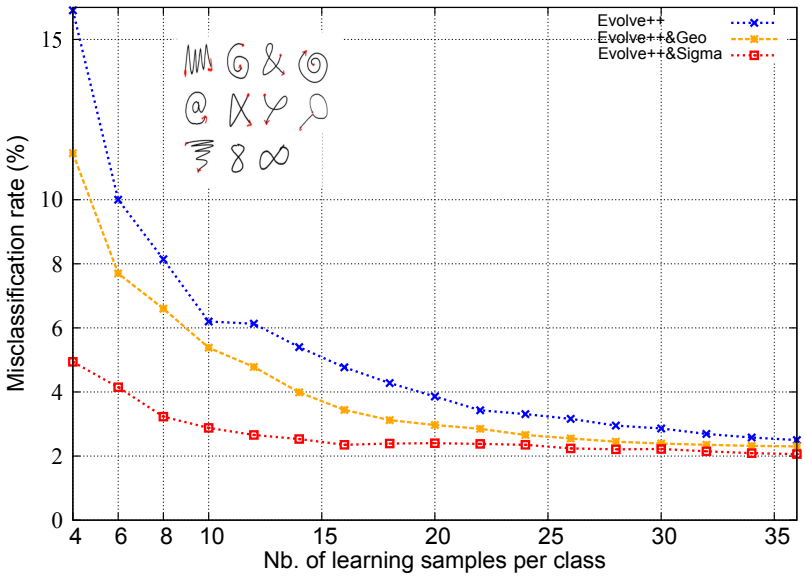



Figure 5.10: Performance improvement when using synthetic data generation in the incremental learning process

We note from Figure 5.10 the important impact of the synthetic gestures at the beginning of the incremental learning process. For example, the misclassification rate is reduced by 50% after 10 real samples per class when the learning process is enriched by synthetic samples. We note also from the same figure that applying distortions on the Sigma-lognormal parameters produces realistic variability in the

synthetic gestures compared to direct geometrical distortions. Thanks to the realistic human-like distortions, the synthetic samples have considerable energy of predicting the future real samples, which significantly accelerates the learning process.

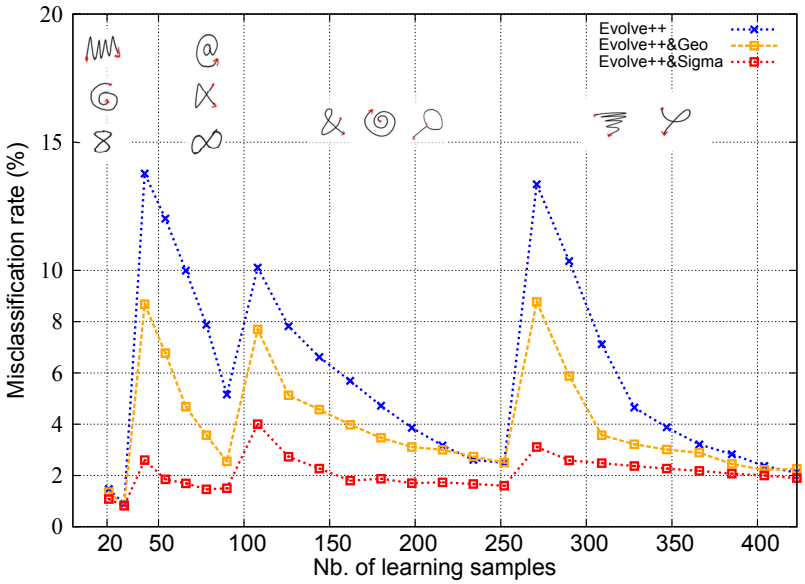



Figure 5.11: Evaluation of the impact of synthetic samples when adding new gestures

Figure 5.11 shows that using synthetic samples, the classifier resists much better when introducing new classes. It is able to rapidly re-estimate all its parameters and to improve the recognition performance for the old and the new gestures. Again, we notice the superiority of lognormal-based deformations over the traditional geometrical ones.

As mentioned before, one advantage of incorporating the lognormal-based generation in the incremental learning process is to obtain an objective numerical evaluation of this handwriting generation technique, instead of the subjective visual evaluation used in precedent works. Therefore, geometrical handwriting generation and motor-based handwriting generation have been compared in the presented figures in a numerical manner that represents the impact of the generated data on the learning process.

In the lognormal-based generation technique that have been used to obtain the results presented in Figures 5.10 and 5.11, random variations are applied on all the sigma-lognormal parameters of each profile (except $t_0$). The variation intervals that we have used are given in Table 5.1 at the last row (strategy E). Nevertheless, we have experimented other strategies in which only some of these parameters are

modified and the others keep unchanged. The five different combinations are listed in Table 5.1. Figure 5.12 shows the corresponding performance obtained by the different combinations of Table 5.1.

We can see that by only applying distortions on the amplitude of each profile $D$ may not be enough to generate realistic and variable gestures. We notice that better results are obtained when coupling D variation with some modifications on the start and the end angle of each stroke. Slight variations of the peripheral parameters $\sigma$ and $\mu$ can introduce useful distortions on the handwritten gesture, as we can note from the curve A in Figure 5.12.

| | $\mu$ | $\sigma$ | D | $\theta_s$ | $\theta_e$ |
|---|---|---|---|---|---|
| Strategy A | $[-5\%, 5\%]$ | $[-5\%, 5\%]$ | - | - | - |
| Strategy B | - | - | $[-5\%, 5\%]$ | - | - |
| Strategy C | $[-5\%, 5\%]$ | $[-5\%, 5\%]$ | $[-5\%, 5\%]$ | - | - |
| Strategy D | - | - | $[-5\%, 5\%]$ | $[-20°, 20°]$ | $[-20°, 20°]$ |
| Strategy E | $[-5\%, 5\%]$ | $[-5\%, 5\%]$ | $[-5\%, 5\%]$ | $[-20°, 20°]$ | $[-20°, 20°]$ |

Table 5.1: The used variation intervals of sigma lognormal parameters and the different tested distortion combinations

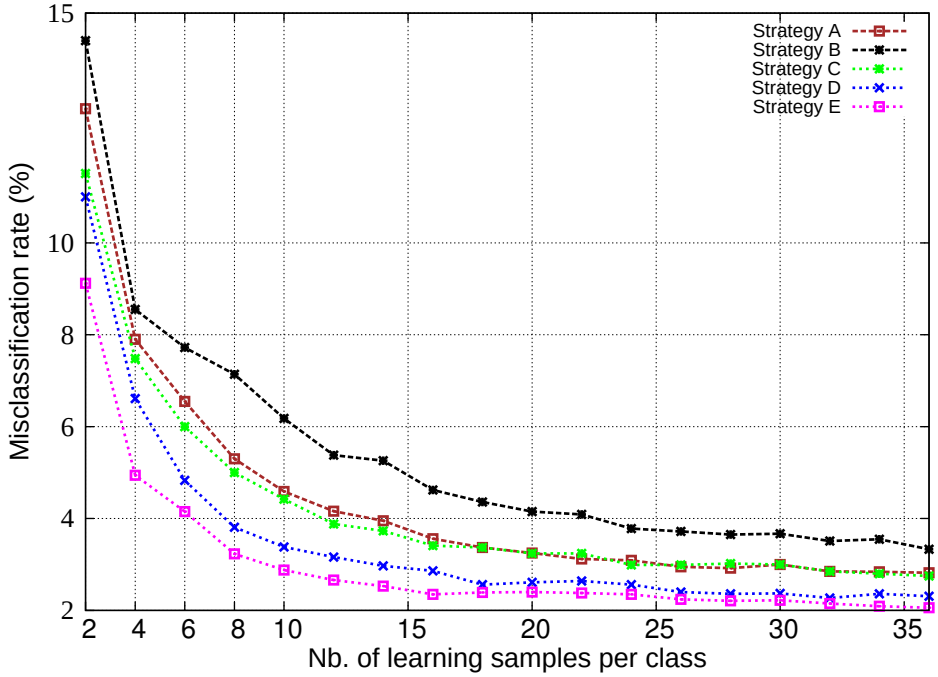



Figure 5.12: Evaluation of distortion combinations presented in Table 5.1

## 5.7 Conclusion

In an online incremental learning of handwriting classifier, the lack of learning data represents an important challenge that can sometime lead to unacceptable performance and make the application difficult to use by the user. Thanks to the Sigma-lognormal representation of handwritten gestures, we incorporated synthetic gestures into the incremental learning process of our evolving handwriting classifier to enhance its prediction capacity. Experimental results show that synthetic samples play a considerable role in improving the classification performance and accelerating the learning process when it starts from scratch and also when new gestures are introduced.

One of our perspectives is to replace the predefined static variation ranges that we used to generate artificial samples by an automatic mechanism that can incrementally estimate an optimal variation interval for each parameter and for each class of gesture. The impact of the variation of the parameter $t_0$ will also be the subject of our near future works.

# Chapter 6

# General Conclusion and Future Work

## 6.1 Conclusion

The purpose of this research work is to go beyond the traditional classification systems in which the set of recognizable categories is predefined at the conception phase and keeps unchanged during its operation. Motivated by the increasing needs of flexible classifiers that can be continuously adapted to cope with its dynamic working environment, we propose in this thesis an evolving classification system and different incremental learning algorithms. The most important characteristic of this system is the ability of learning new classes from few data during the operation phase. Contrary to the traditional batch ones, the lifecycle of evolving classifiers does not comprise independent conception (learning) and operation phases. The learning process of these evolving approaches is performed in an incremental and lifelong manner. They continuously profit from any available data to adjust its knowledge base, and more importantly, to learn new unseen classes.

Evolving classifiers are very useful in many application areas, such as face recognition, customer profiling, etc. In these applications, new classes (categories) can either be discovered from the incoming data, or explicitly added by the user. However, our principal applicative motivation is related to online handwritten gesture recognition systems. These systems must be able to deal with the changing user-specific needs and behavior. The user must be allowed to extend the functionalities of the recognition system by adding new classes (gestures), and the system must offer continuous adaptation to the manner in which the user draws the gestures. Therefore, motivated by the high interaction between the user and the system in this context, and the increasing needs for dynamic and adaptive classification approaches, we have focused our research on the online incremental learning of evolving classification systems.

Our classification approach is based on Takagi-Sugeno (TS) fuzzy rule-based architecture. A TS model consists of two parts: non-linear antecedents and linear consequent functions. We propose in our system a new antecedent structure that takes into consideration the correlations between the axes of the input space (the features). We present the recursive estimation formulas required to incrementally learn this new structure.

We have placed a special focus in the learning algorithms on the stability of consequent learning in spite of the evolving antecedent structure. We have dealt with

this issue in two different manners. A memory buffer is used in the first solution to allow a temporary stability in the antecedent structure and thus better consequent estimation. As a second solution, we have proposed a novel learning paradigm in which the antecedent learning is controlled (and minimized) based on a feedback signal from the system output. More precisely, the antecedent modifications are reinforced for the data samples that show a high risk of misclassification, and considerably reduced for the other samples. Based on this concept, we have proposed the Evolve++ algorithm: a confusion-driven incremental learning for evolving TS classification systems.

Our evolving classifier with the enhanced antecedent structure and the two learning methods has shown very good results in the experiments that have been conducted on several benchmarks. It has been compared to a well-known evolving TS model that had been the object of many significant publications. As we have seen in the fourth chapter (experimental validation), our system can rapidly achieve high recognition rates when learning from scratch a set of classes in synchronized manner. Moreover, it shows high resistance and fast recovery when adding new classes in progressive (unsynchronized) scenario.

Although our evolving classification approach can be applied to many applicative contexts, we have placed a particular focus on the field of online handwriting recognition which had been the original applicative motivation of the abovementioned theoretical contribution. The goal is to obtain customizable handwritten gesture recognition systems that can be used, for example, to interpret handwritten shortcuts in touch screen computers. Using theses evolving handwriting classifiers, the user will be able to add new gestures to the recognition systems by only provided few samples. Even though our evolving classifier had shown good results when tested on a dataset of handwritten gestures, we have taken a step ahead in performance improvement by integrating synthetic gesture generation technique in the incremental learning process. We aim by this generation technique at overcoming the problem of lack of data at the beginning of learning new gestures. The generation is performed by applying some deformations on the real samples in realistic and human-like manner. Instead of the traditional geometrical deformations used in related works, we applied distortions on the motor representation space of the handwritten gestures. Encouraged by its numerous theoretical and experimental proofs, we have chosen the Sigma-lognormal modeling as a new representation space of the

handwritten gestures. It has been experimentally demonstrated in the fifth chapter that the learning of evolving handwriting classifiers can be considerably accelerated thanks to synthetic data. The results have also shown that samples generated by lognormal-based distortions have more positive impact on the learning, which strongly suggests that the lognormal-based introduced variability is wider and more realistic than direct geometrical one.

## 6.2 Future works

Similar to the contributions presented in this thesis, our perspectives can be divided into two categories: theoretical perspectives and application-related perspectives.

As theoretical (application-independent) future works, we present below the direct important tracks that we will examine to improve our approach:

- A new solution to cope with consequent learning stability in evolving TS models is to incorporate after each antecedent modification an adjustment term into the Recursive Least Squares process. This adjustment term must affect the autocorrelation matrix $\Phi^{-1}$ in order to keep, as much as possible, a correct accumulated cost function in the consequent optimization. The adjustment vector must be defined based on formal mathematical relationship between antecedent modifications and the knowledge accumulated in the recursive consequent learning. A deep investigation of this formal solution is one of our perspectives.

- One of the important topics that we will study is the capacity of deleting existing classes. This functionality should be available in the evolving classifier in order to get rid of unused classes. The deletion must be done in an efficient and transport manner.

- In addition to the first two points, one of the interesting topics that we would like to approach is the Incremental feature selection and extraction methods (see Section 2.6). Evolving data representation space can be coupled to evolving classification systems to obtain powerful dynamic systems.

- Besides the abovementioned technical ideas, we are planning to do some efforts to facilitate comparisons between evolving classifiers and incremental learning

algorithms. Due to the absence of benchmarks in the field of evolving classification systems, we contemplate proposing a public benchmark that covers the different aspects in this domain. In addition to public multi-class datasets, we will propose a number of tasks that represent different experimental scenarios in which the classes are learned in either synchronized or unsynchronized manner. A complete description of these scenarios should be presented in order to facilitate easy and precise comparisons between evolving classification approaches. Moreover, official competitions can be organized based on these benchmark tasks.

- A special focus can be placed in the medium-term on a more sophisticated antecedent structure using membership functions of type-2 fuzzy sets [76]. This new concept can lead to more powerful antecedent structure by handling more uncertainty than that of traditional type-1 sets. The recursive estimation of type-2 membership functions and the impact of this antecedent structure on the system performance represent two interesting research topic that will be considered in future works.

In addition to the above-mentioned perspectives that are not related to a specific application area, we would like to take new steps ahead in the development of evolving handwriting classifiers. Several points stand as future working tracks in the short and the medium-term. We present here the most important points:

- We will continue our interesting joint work with Scribens laboratory concerning handwriting generation based on Sigma-lognormal modeling. The near future work will be focused on proposing an incremental learning mechanism of the parameter variation intervals. The handwriting generation approach presented in the precedent chapter is class-independent; i.e. the applied distortions (and the predefined variation intervals) are general and not related to specific gestures. Our idea is to keep applying these general variation intervals for the first few samples of a new introduced gesture, but with adjusting them for each class of gesture according to the nature deformation observed between the real samples of this specific gesture. A variance/covariance matrix of sigma-lognormal parameters can be incrementally estimated for each gesture and then replaces the predefined intervals in synthetic data generation.

Moreover, statistical analysis (such as principal component analysis) of the estimated matrices can be carried out in order to discover potential relationships between the variations in the Sigma-lognormal parameters and the writer physical and health conditions.

- We will carry out more investigations about the incorporation of our evolving handwritten gesture classifier in real contexts. A first track consists in working with Synchromedia laboratory for the incorporation of our evolving recognition system in sensitive collaborative annotation interfaces. We would like to study the different challenges the might appear in this new applicative contexts.

  Another important aspect that must be studied is the user interaction with the recognition systems. In the experiments conducted in this thesis, the tests had been performed on a gesture dataset that had been collected in passive mode (no real interactions with the user). In real systems, the user usually makes some adjustments on his behavior (or data) in order to favorite correct classification response and to avoid potential errors. Therefore, mutual adaptation phenomena must be studied when using evolving systems in online and interactive manner, which is the case of evolving handwritten gesture classifier. Although this mutual adaptation is generally supposed to have positive effects on the global system performance, we have to verify that the continuous system learning and adaption are performed in suitable manner to avoid potential perturbations at the user side.

  To lead an accurate study about the abovementioned mutual adaptation aspect, we will work in collaboration with CRPCC research group from LOUSTIC laboratory. CRPCC members are specialized in experimental psychology and ergonomics of conception of human-machine interfaces. The experiments will be conducted in real-context using the different applications (Section 4.5) into which we have incorporated our evolving classification system.

# Bibliography

[1] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas, "Experiments with adaptation strategies for a prototype-based recognition system for isolated handwritten characters," *International Journal on Document Analysis and Recognition*, vol. 3, pp. 150–159, 2001.

[2] J. S. R. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEE Transactions On Systems Man And Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[3] H. Mouchere, E. Anquetil, and N. Ragot, "Writer style adaptation in online handwriting recognizers by a fuzzy mechanism approach : The adapt method," *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, vol. 21, no. 1, pp. 99–116, 2007.

[4] R. C. Chakraborty, "Adaptive resonance theory: Soft computing course lecture," Decembre 2010.

[5] N. Kasabov and Q. Song, "Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144 –154, Apr. 2002.

[6] S. Lühr and M. Lazarescu, "Incremental clustering of dynamic data streams using connectivity based representative points," *Data and Knowledge Engineering*, vol. 68, no. 1, pp. 1 – 27, 2009.

[7] E. Lughofer, "Flexfis: A robust incremental learning approach for evolving takagi-sugeno fuzzy models," *IEEE Transactions On Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[8] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 769 –783, 2000.

[9] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *Fuzzy Systems, IEEE Transactions on*, vol. 16, no. 6, pp. 1462 –1475, dec. 2008.

[10] A. Cornuéjols, "Getting order independence in incremental learning," in *Proc. European Conference on Machine Learning 1993, (P.B. Brazdil, Ed.), Lecture Notes in Artificial Intelligence 667*. Springer-Verlag, 1993, pp. 196–212.

[11] H. Mitoma, S. Uchida, and H. Sakoe, "Online character recognition based on elastic matching and quadratic discrimination," in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ser. ICDAR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 36–40. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2005.178

[12] J. Wang, C. Wu, Y.-Q. Xu, and H.-Y. Shum, "Combining shape and physical modelsfor online cursive handwriting synthesis," *International Journal on Document Analysis and Recognition*, vol. 7, pp. 219–227, September 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1102243.1102244

[13] S. Uchida and H. Sakoe, "A Survey of Elastic Matching Techniques for Handwritten Character Recognition," *IEICE Transactions on Information and Systems*, vol. E88-D, no. 8, pp. 1781–1790, 2005.

[14] Z. Lin and L. Wan, "Style-preserving english handwriting synthesis," *Pattern Recognition*, vol. 40, pp. 2097–2109, 2007.

[15] H. Mouchère, S. Bayoudh, E. Anquetil, and L. Miclet, "Synthetic on-line handwriting generation by distortions and analogy," in *13th Conference of the International Graphonomics Society (IGS2007)*, November 2007, pp. 10–13.

[16] R. Plamondon, M. Djioua, and C. O'Reilly, "La théorie cinématique des mouvements humains rapides: développements récents," *Traitement du signal*, no. 5, 2009.

[17] P. Mitra, S. Member, C. A. Murthy, and S. K. Pal, "A probabilistic active support vector learning algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, p. 2004, 2004.

[18] L. Zhang, C. Chen, J. Bu, D. Cai, X. He, and T. S. Huang, "Active learning based on locally linear reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. PrePrints, 2011.

[19] P.-H. Gosselin and M. Cord, "Active learning methods for Interactive Image Retrieval." *IEEE Transactions on Image Processing.*, vol. 17, no. 7, pp. 1200–1211, 2008.

[20] M. A. Maloof and R. S. Michalski, "Incremental learning with partial instance memory," *Artif. Intell.*, vol. 154, pp. 95–126, April 2004.

[21] R. E. Reinke and R. S. Michalski, *Incremental learning of concept descriptions: A method and experimental results.* New York, NY, USA: Oxford University Press, Inc., 1988, pp. 263–288. [Online]. Available: http://portal.acm.org/citation.cfm?id=60769.60780

[22] N. Littlestone, "Redundant noisy attributes, attribute errors, and linear-threshold learning using winnow," in *COLT '91: Proceedings of the fourth annual workshop on Computational learning theory.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pp. 147–156.

[23] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, no. 1, pp. 37–66, 1991.

[24] R. Polikar, L. Udpa, S. Udpa, S. Member, S. Member, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Transactions on System, Man and Cybernetics*, vol. 31, pp. 497–508, 2001.

[25] J. Zwickel and A. J. Wills, *New Directions in Human Associative Learning.* Psychology Press, 2005, ch. Integrating associative models of supervised and unsupervised categorization, p. 118.

[26] J. J. L. Jr. and R. C. Zeleznik, "A practical approach for writer-dependent symbol recognition using a writer-independent symbol recognizer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1917–1926, 2007.

[27] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*, 2nd ed. Wiley-Interscience, Nov. 2001. [Online]. Available: http://www.worldcat.org/isbn/0471056693

[28] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Art 2-a: An adaptive resonance algorithm for rapid category learning and recognition," *Neural Networks*, vol. 4, pp. 493–504, 1991.

[29] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464 –1480, Sep. 1990.

[30] T. M. Martinetz and K. J. Shulten, "A "neural-gas" network learns topologies"," in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds., 1991, pp. 397–402.

[31] E. Lughofer, *Evolving fuzzy models: incremental learning, interpretability, and stability issues, applications.* VDM Verlag Dr. M
"uller, 2008.

[32] P. Foggia, G. Percannella, C. Sansone, and M. Vento, "A graph-based clustering method and its applications," in *Advances in Brain, Vision, and Artificial Intelligence*, ser. Lecture Notes in Computer Science, F. Mele, G. Ramella, S. Santillo, and F. Ventriglia, Eds. Springer Berlin / Heidelberg, 2007, vol. 4729, pp. 277–287.

[33] J. Choo, R. Jiamthapthaksin, C. sheng Chen, O. U. Celepcikay, C. Giusti, and C. F. Eick, "Mosaic: A proximity graph approach for agglomerative clustering," in *In: The 9th Intl. Conf. on Data Warehousing and Knowledge Discovery*, 2007.

[34] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68 –75, Aug. 1999.

[35] R. R. Yager and D. P. Fileu, "Learning of fuzzy rules by mountain clustering," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, B. Bosacchi & J. C. Bezdek, Ed., vol. 2061, Dec. 1993, pp. 246–254.

[36] P. Angelov and D. Filev, "An approach to online identification of takagi-sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 1, pp. 484–498, Feb. 2004.

[37] F. Minku, H. Inoue, and X. Yao, "Negative correlation in incremental learning," *Natural Computing*, vol. 8, pp. 289–320, 2009.

[38] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn++.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes." *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 152–168, 2009.

[39] S. Mitaim and B. Kosko, "The shape of fuzzy sets in adaptive function approximation," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 637 –656, aug 2001.

[40] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338 – 353, 1965. [Online]. Available: http://www.sciencedirect.com/science/article/B7MFM-4DX43MN-W3/2/f244f7a33f31015e819042700cd83047

[41] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. SMC-3, no. 1, pp. 28 –44, jan. 1973.

[42] E. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, vol. 26, pp. 1182–1191, 1977.

[43] W. L. Ho, W. L. Tung, and C. Quek, "An evolving mamdani-takagi-sugeno based neural-fuzzy inference system with improved interpretability-accuracy," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, july 2010, pp. 1 –8.

[44] J.-S. Jang and C.-T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *Neural Networks, IEEE Transactions on*, vol. 4, no. 1, pp. 156 –159, jan 1993.

[45] K. Hunt, R. Haas, and R. Murray-Smith, "Extending the functional equivalence of radial basis functionnetworks and fuzzy inference systems," *IEEE Transactions on Neural Networks*, vol. 7, no. 3, pp. 776–781, May 1996. [Online]. Available: http://eprints.gla.ac.uk/2927/

[46] E. Lughofer, "Extensions of vector quantization for incremental clustering," *Pattern Recognition*, vol. 41, no. 3, pp. 995–1011, 2008.

[47] P. Angelov and E. Lughofer, "Data-driven evolving fuzzy systems using ets and flexfis: comparative analysis," *International Journal of General Systems*, vol. 37, pp. 45–67, 2008.

[48] N. Kasabov, "Evolving fuzzy neural networks: Theory and applications for on-line adaptive prediction, decision making and control," in *Control. Australian Journal of Intelligent Information Processing Systems 5*, 1998, pp. 154–160.

[49] S. Salzberg, "A nearest hyperrectangle learning method," *Machine Learning*, vol. 6, pp. 251–276, 1991.

[50] F. Ferrer-Troyano, J. Aguilar-Ruiz, and J. Riquelme, "Incremental rule learning and border examples selection from numerical data streams," *Journal of Universal Computer Science*, vol. 11, no. 8, pp. 1426–1439, 2005.

[51] W. Tung and C. Quek, "A mamdani-takagi-sugeno based linguistic neural-fuzzy inference system for improved interpretability-accuracy representation," in *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, aug. 2009, pp. 367 –372.

[52] H. Ying, Y. Ding, S. Li, and S. Shao, "Typical takagi-sugeno and mamdani fuzzy systems as universal approximators: necessary conditions and comparison," in *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, vol. 1, may 1998, pp. 824 –828 vol.1.

[53] S. Ozawa, S. L. Toh, S. Abe, S. Pang, and N. Kasabov, "Incremental learning of feature space and classifier for face recognition," *Neural Networks*, vol. 18, no. 5-6, pp. 575 – 584, 2005, iJCNN 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6T08-4GX1J0V-3/2/f67bae3b2576ae7c809e0b462b26394e

[54] S. Pang, S. Ozawa, and N. Kasabov, "Incremental linear discriminant analysis for classification of data streams," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 5, pp. 905 –914, 2005.

[55] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[56] G. Moustakides, "Study of the transient phase of the forgetting factor rls," *Signal Processing, IEEE Transactions on*, vol. 45, no. 10, pp. 2468 –2476, Oct. 1997.

[57] "http://www.synchromedia.ca/web/ets/gesturedataset," 2010.

[58] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009. [Online]. Available: http://dx.doi.org/10.1145/1656274.1656278

[59] P. Angelov, E. Lughofer, and X. Zhou, "Evolving fuzzy classifiers using different model architectures," *Fuzzy Sets and Systems*, vol. 159, no. 23, pp. 3160 – 3182, 2008, theme: Modeling. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165011408003333

[60] A. Ghorbel, A. Almaksour, A. Lemaitre, and E. Anquetil, "Incremental learning for interactive sketch recognition," in *Nineth IAPR International Workshop on Graphics RECognition (GREC 2011)*, 2011.

[61] S. Uchida and H. Sakoe, "Eigen-Deformations for Elastic Matching based Handwritten Character Recognition," *Pattern Recognition*, vol. 36, no. 9, pp. 2031–2040, Sep. 2003.

[62] S. Bayoudh, L. Miclet, H. Mouchère, and E. Anquetil, "Learning a classifier with very few examples: Analogy based and knowledge based generation of new

examples for character recognition," in *18th European Conference on Machine Learning (ECML)*, Warsaw, Poland, September 2007.

[63] H. Bezine, M. Kefi, and A. M. Alimi, "On the beta-elliptic model for the control of the human arm movement," *IJPRAI*, vol. 21, no. 1, pp. 5–19, 2007.

[64] P. Morasso and F. Mussa-Ivaldi, "Trajectory formation and handwriting: A computational model," *Biological Cybernetics*, vol. 45, no. 2, pp. 131–142, 1982.

[65] F. Leclerc, R. Plamondon, and L. G., "Gaussian curves for signature modelization and word segmentation," *Traitement du signal*, vol. 9, no. 4, pp. 347–358, 1992.

[66] G. Gangadhar, D. Joseph, and V. S. Chakravarthy, "An oscillatory neuromotor model of handwriting generation," *International Journal on Document Analysis and Recognition*, vol. 10, pp. 69–84, November 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1327266.1327270

[67] R. Plamondon, "A kinematic theory of rapid human movements," *Biological Cybernetics*, vol. 72, no. 4, pp. 295–307, 1995.

[68] F. Lacquaniti, C. Terzuolo, and P. Viviani, "The law relating the kinematic and figural aspects of drawing movements," *Acta Psychologica*, vol. 54, pp. 115–130, 1983.

[69] P. Morasso, F. Ivaldi, and C. Ruggiero, "How a discontinuous mechanism can produce continuous patterns in trajectory formation and handwriting," *Acta Psychologica*, vol. 54, no. 1-3, pp. 83 – 98, 1983. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0001691883900252

[70] R. Plamondon, A. M. Alimi, P. Yergeau, and F. Leclerc, "Modelling velocity profiles of rapid movements: a comparative study," *Biological Cybernetics*, vol. 69, no. 2, pp. 119–128, 1993.

[71] R. Plamondon, "A kinematic theory of rapid human movements: Part iii. kinetic outcomes," *Biological Cybernetics*, vol. 78, no. 2, pp. 133–145, 1998.

[72] R. Plamondon and M. Djioua, "A multi-level representation paradigm for handwriting stroke generation," *Human Movement Science*, vol. 25,

no. 4-5, pp. 586 – 607, 2006, advances in Graphonomics: Studies on Fine Motor Control, Its Development and Disorders. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167945706000613

[73] M. Djioua and R. Plamondon, "Studying the variability of handwriting patterns using the kinematic theory," *Human Movement Science*, vol. 28, no. 5, pp. 588–601, 2009.

[74] C. O'Reilly and R. Plamondon, "Development of a sigma-lognormal representation for on-line signatures," *Pattern Recognition*, vol. 42, no. 12, pp. 3324–3337, 2009.

[75] M. Djioua and R. Plamondon, "An interactive system for the automatic generation of huge handwriting databases from a few specimens," in *ICPR*, 2008, pp. 1–4.

[76] H. Wu and J. Mendel, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *Fuzzy Systems, IEEE Transactions on*, vol. 10, no. 5, pp. 622 – 639, oct 2002.

# Author bibliography


[AA11b] Abdullah Almaksour and Eric Anquetil. Improving premise structure in evolving Takagi-Sugeno neuro-fuzzy classifiers. *Evolving Systems Journal*, 2(1):25–33, 2011.

[AA11a] A. Almaksour and E. Anquetil. Systèmes dinférence floue auto-évolutifs : applications sur lapprentissage incrémental de systèmes de reconnaissance de gestes manuscrits. *Revue Document Numérique*, 14(2), 2011.

[AMA09] A. Almaksour, H. Mouchère, and E. Anquetil. Apprentissage incrémental avec peu de données pour la reconnaissance de caractères manuscrits en-ligne. *Revue Traitement du Signal*, 26(5):323–338, 2009.

[AEPO11] Abdullah Almaksour, Anquetil Eric, Réjean Plamondon, and Christian O'Reilly. Synthetic handwritten gesture generation using sigma-lognormal model for evolving handwriting classifiers. In *Proceedings of 15th Biennial Conference of the International Graphonomics Society (IGS 2011)*, 2011.

[AA10a] Abdullah Almaksour and Eric Anquetil. Improving premise structure in evolving takagi-sugeno neuro-fuzzy classifiers. In *Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA 2010)*, 2010.

[AAQC10b] Abdullah Almaksour, Eric Anquetil, Solen Quiniou, and Mohammed Cheriet. Personalizable pen-based interface using lifelong learning. In *Proceedings of the Twelfth International Conference on Frontiers in Handwriting Recognition (ICFHR'10)*, pages 188–193, 2010.

[AAQC10a] Abdullah Almaksour, Eric Anquetil, Solen Quiniou, and Mohammed Cheriet. Evolving fuzzy classifiers: Application to incremental learning of handwritten gesture recognition systems. In *International Conference on Pattern Recognition (ICPR 2010)*, 2010.

[AA09] Abdullah Almaksour and Eric Anquetil. Fast incremental learning strategy driven by confusion reject for online handwriting recognition. In *Tenth International Conference on Document Analysis and Recognition (ICDAR 2009)*, pages 81–85, 2009.

[AMA08b] Abdullah Almaksour, Harold Mouchère, and Eric Anquetil. Fast online incremental learning with few examples for online handwritten character recognition. In *Proceedings of the Eleventh International Conference on Frontiers*

*in Handwriting Recognition (ICFHR'08)*, pages 623–628, Montréal, Québec, Canada, August 2008.

[AA10b] Abdullah Almaksour and Eric Anquetil. A robust learning algorithm for evolving first-order takagi-sugeno fuzzy classifiers. In *Conférence Francophone sur l'Apprentissage Automatique (CAP 2010)*, pages 33–48, 2010.

[AA10c] Abdullah Almaksour and Eric Anquetil. Systèmes dinférence floue auto-évolutifs : applications sur lapprentissage incréèmental de systèmes de reconnaissance de gestes manuscrits. In *Actes du XIème Colloque International Francophone sur l'Ecrit et le Document (CIFED'10)*, pages 233–247, 2010.

[AMA08a] Abdullah Almaksour, Harold Mouchère, and Eric Anquetil. Apprentissage incrémental et synthèse de données pour la reconnaissance de caractères manuscrits en-ligne. In *Actes du Xème Colloque International Francophone sur l'Ecrit et le Document (CIFED'08)*, volume 1, pages 55–60, 2008.

[GALA11] Achraf Ghorbel, Abdullah Almaksour, Aurélie Lemaitre, and Eric Anquetil. Incremental learning for interactive sketch recognition. In *Nineth IAPR International Workshop on Graphics RECognition (GREC 2011)*, 2011.