NOVA IMS

The ENFS-Uni0-Evolving Classifier: Mathematical Formulation and Conceptual Foundations

–

Prof: Paulo Vitor de Campos Souza

Models Explanations

Student:	————-
ID:	220
Date:	February 01, 2026

Contents

Chapter 1

The ENFS-Uni0-Evolving Classifier

This chapter presents the **ENFS-Uni0-Evolving** [1], a fully online evolving neuro-fuzzy classifier derived from the original ENFS-Uni0 model proposed by Souza and Lughofer (2021). The version implemented and evaluated in this dissertation includes several conceptual and computational extensions:

- an ADPA-based dynamic fuzzification layer;
- evolving fuzzy rules represented as *data clouds*;
- an OR-type aggregation neuron (probabilistic t-conorm);
- incremental feature relevance learning (Lughofer-style);
- fully recursive RLS for the consequent layer with numerical stabilizers;
- interpretable tracking of rule evolution, feature relevance, and drift regions.

This chapter provides the full mathematical formulation, organized per layer, and highlights the original contributions introduced in this work.

1.1 Notation and Preliminaries

We consider a data stream

$$(\mathbf{x}_1, y_1),\ (\mathbf{x}_2, y_2),\ \ldots, (\mathbf{x}_t, y_t), \ldots$$

with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \{0, 1\}$ (extension to C classes is straightforward). At each time step the model updates its fuzzy rules, activation strengths, consequent parameters and RLS statistics.

Throughout the chapter we use the following notation:

- $\mathcal{R}(t)$ – the set of fuzzy rules available at time t.
- $R(t) = |\mathcal{R}(t)|$ – the number of rules at time t. This quantity evolves dynamically as ADPA creates, adapts or prunes rules.
- $\mathbf{c}_r \in \mathbb{R}^d$ – the center (prototype) of rule r, obtained via ADPA clustering.
- $\sigma_r > 0$ – the rule radius (Gaussian width) determining the spread of membership functions around $\mathbf{c}_r$.
- $\mathbf{w}_r = [w_{r,1}, \dots, w_{r,d}]^\top$ – the feature–relevance vector of rule r. These weights are updated incrementally following Lughofer's relevance–learning mechanism and normalised so that $\sum_j w_{r,j} = 1$.
- $g_{1,r}, g_{2,r}, z_r$ – UniNull–uninorm parameters used in the original ENFS-Uni0 model. They are kept in this evolving variant for backward compatibility and potential future extensions, although aggregation is performed through a stable OR-type t-conorm in this work.
- $\mathbf{v}_r \in \mathbb{R}^C$ – the local class–certainty vector traditionally used in the original ENFS-Uni0. In the present model, its role is absorbed into the global RLS matrix $\mathbf{W}$ (see below) to improve numerical stability.
- $\boldsymbol{\phi}_t \in \mathbb{R}^{R(t)+1}$ – the activation vector at time t, defined as

  $$\boldsymbol{\phi}_t = [1,\ y_1(\mathbf{x}_t), \dots, y_{R(t)}(\mathbf{x}_t)]^\top,$$

  where $y_r(\mathbf{x}_t)$ denotes the activation strength (firing level) of rule r at time t.
- $\mathbf{W}_t \in \mathbb{R}^{(R(t)+1)\times C}$ – the global consequent parameter matrix updated by Recursive Least Squares (RLS). It maps rule activations to class scores:

  $$\mathbf{s}_t = \mathbf{W}_t^\top \boldsymbol{\phi}_t.$$

- $\mathbf{P}_t \in \mathbb{R}^{(R(t)+1)\times(R(t)+1)}$ – the RLS covariance (inverse correlation) matrix controlling learning speed and numerical stability.
- $\boldsymbol{\mu}_t$ – global running mean of the data stream.
- X_t – running average of squared norms: $X_t = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i\|^2$.

- VC_t – global variance characteristic used by ADPA to determine the spread of the stream:
$$VC_t = \sqrt{\max\Big(2(X_t - \|\boldsymbol{\mu}_t\|^2),\, \varepsilon\Big)}.$$
- $\Sigma_t = VC_t/2$ – global radius used for rule creation and adaptation inside ADPA.
- s_r – support of rule r, i.e., number of samples that have been assigned (adapted) to this rule.
- θ – similarity threshold for pruning highly redundant rules.
- $R_{\max}$ – optional upper bound on the number of rules.

This notation provides a complete map of all variables used in the ENFS-Uni0-Evolving model and will be referenced consistently throughout the following sections.

1.2 Layer 1 – Dynamic Fuzzification via ADPA

The first layer of ENFS-Uni0-Evolving performs *dynamic fuzzification* through the Autonomous Data Partitioning Algorithm (ADPA). Instead of pre-defining a fixed grid of membership functions in each input dimension, the model incrementally discovers *data clouds* in the joint feature space. Each data cloud becomes the antecedent of a fuzzy rule.

This strategy is particularly suitable for non-stationary software metrics: regions of the feature space that are never visited by the stream are never assigned rules, while new regions that emerge later in the project (e.g., because of process changes or new contributor profiles) can be covered by newly created rules.

Formally, at time t the algorithm maintains global statistics:

1. **Global mean**
$$\boldsymbol{\mu}_t = \frac{1}{t}\sum_{i=1}^{t} \mathbf{x}_i, \tag{1.1}$$
2. **Second-moment estimate**
$$X_t = \frac{1}{t}\sum_{i=1}^{t} \|\mathbf{x}_i\|^2, \tag{1.2}$$
3. **Global variance characteristic**
$$VC_t = \sqrt{\max\Big(2\Big(X_t - \|\boldsymbol{\mu}_t\|^2\Big), \epsilon\Big)}, \tag{1.3}$$

4. **Global radius**

$$\Sigma_t = \frac{VC_t}{2}. \tag{1.4}$$

The scalar Σ_t acts as an adaptive radius summarizing the current spread of the stream. Existing rule centres $\mathbf{c}_r$ are interpreted as local representatives of the empirical distribution. The Chebyshev distance

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i| \tag{1.5}$$

is used because it is computationally simple, scale-robust and emphasises worst-case deviations in any dimension, which is desirable when tracking critical software-process indicators.

A new rule is created whenever the incoming sample is "structurally incompatible" with the current set of centres:

$$\min_r d_\infty(\mathbf{c}_r, \boldsymbol{\mu}_t) > d_\infty(\mathbf{x}_t, \boldsymbol{\mu}_t) \quad \text{or} \quad \max_r d_\infty(\mathbf{c}_r, \boldsymbol{\mu}_t) < d_\infty(\mathbf{x}_t, \boldsymbol{\mu}_t). \tag{1.6}$$

Intuitively, the sample either falls in a region closer to the global mean than any existing centre (a "new typical pattern"), or farther away than all centres (a "new extreme regime").

Otherwise, the closest rule is adapted:

$$\mathbf{c}_r \leftarrow \frac{s_r \mathbf{c}_r + \mathbf{x}_t}{s_r + 1}, \qquad \sigma_r \leftarrow 0.8\,\sigma_r + 0.2\,\Sigma_t, \tag{1.7}$$

where s_r is the rule support (number of assigned samples). Thus, centres follow the stream by exponential smoothing and the rule radius σ_r slowly tracks the global spread, preserving some memory of past data.

1.3 Layer 2 – Gaussian Memberships and OR Aggregation

Once rule centres and radii are defined, fuzzification in each rule is implemented by Gaussian membership functions:

$$a_{r,j}(x_j) = \exp\left(-\frac{(x_j - c_{r,j})^2}{2\sigma_r^2}\right), \quad j = 1, \ldots, d. \tag{1.8}$$

Each $a_{r,j}(x_j)$ expresses how compatible feature j is with the local data cloud of rule r.

In the original ENFS-Uni0, these memberships are combined by a UniNull neuron based on uninorms, allowing smooth transitions between AND-like and OR-like behaviour. In the present evolving variant, to improve numerical stability in long streams and to obtain a more direct interpretation, we adopt an **OR-type aggregation** reminiscent of probabilistic sum:

$$y_r(\mathbf{x}_t) = \frac{\sum_{j=1}^{d} w_{r,j}(t)\, a_{r,j}(x_{t,j})}{\sum_{j=1}^{d} w_{r,j}(t) + \varepsilon}, \tag{1.9}$$

where $\varepsilon > 0$ avoids division by zero.

Equation (1.9) can be read as: *"rule r fires strongly if at least one highly relevant attribute has high membership in the rule's local cloud".* This matches well the semantics of anomaly detection in software projects, where a strong indication on a single key metric (e.g., large delay days or sudden growth in issue churn) can be sufficient to trigger a rule.

1.3.1 Incremental Feature Relevance Weights

To preserve interpretability and adaptivity, ENFS-Uni0-Evolving does not use fixed or random relevance weights. Instead, following Lughofer's incremental relevance learning, each $w_{r,j}(t)$ is updated according to the local contribution of feature j whenever rule r is activated.

A simplified update scheme is:

$$w_{r,j}(t+1) = (1 - \eta_w)\, w_{r,j}(t) + \eta_w\, a_{r,j}(x_{t,j}), \tag{1.10}$$

where $\eta_w \in (0, 1)$ is a learning rate. The term $a_{r,j}(x_{t,j})$ acts as a data-driven relevance indicator: features that frequently exhibit high membership when rule r fires will see their weights reinforced over time.

To maintain a relative scale across features, the weights are renormalised:

$$w_{r,j}(t+1) \leftarrow \frac{w_{r,j}(t+1)}{\sum_{k=1}^{d} w_{r,k}(t+1) + \varepsilon}. \tag{1.11}$$

This mechanism provides a direct link between data and interpretable knowledge:

- high $w_{r,j}(t)$ means that feature j is structurally important for rule r at time t;

- temporal trajectories $t \mapsto w_{r,j}(t)$ reveal how the importance of a metric changes along the life-cycle of a software project.

1.4 Layer 3 – Consequent Layer with Recursive Least Squares

The third layer implements a global linear mapping from rule activations to class scores, trained online by Recursive Least Squares (RLS). Let the activation vector be

$$\boldsymbol{\phi}_t = \Big[1,\ y_1(\mathbf{x}_t), \dots, y_{R(t)}(\mathbf{x}_t)\Big]^\top \in \mathbb{R}^{R(t)+1}, \tag{1.12}$$

where the first component is a bias term and the remaining components are rule activations.

Class scores are computed as:

$$\mathbf{s}_t = \mathbf{W}_t^\top \boldsymbol{\phi}_t, \tag{1.13}$$

with $\mathbf{W}_t \in \mathbb{R}^{(R(t)+1)\times C}$, where C is the number of classes (two in the delay/no-delay problem). The predicted class is

$$\hat{y}_t = \arg\max_c s_{t,c}. \tag{1.14}$$

1.4.1 RLS Update and Stability Mechanisms

Given the one-hot target vector $\mathbf{d}_t$ (with $(\mathbf{d}_t)_c = 1$ for $c = y_t$ and zero otherwise), the RLS update is:

$$\mathbf{K}_t = \frac{\mathbf{P}_{t-1}\boldsymbol{\phi}_t}{\lambda + \boldsymbol{\phi}_t^\top \mathbf{P}_{t-1}\boldsymbol{\phi}_t}, \tag{1.15}$$

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \mathbf{K}_t\Big(\mathbf{d}_t - \mathbf{W}_{t-1}^\top \boldsymbol{\phi}_t\Big)^\top, \tag{1.16}$$

$$\mathbf{P}_t = \frac{1}{\lambda}\Big(\mathbf{P}_{t-1} - \mathbf{K}_t\boldsymbol{\phi}_t^\top \mathbf{P}_{t-1}\Big), \tag{1.17}$$

where $\lambda \in (0, 1]$ is the forgetting factor and $\mathbf{P}_t$ is the inverse covariance estimate.

In long data streams, numerical issues may arise if the denominator $\lambda + \boldsymbol{\phi}_t^\top \mathbf{P}_{t-1}\boldsymbol{\phi}_t$ becomes too small or if accumulated rounding errors destabilise $\mathbf{P}_t$. The implemented version therefore includes:

- a lower bound on the denominator:

$$\text{denom} \leftarrow \max\left(\lambda + \boldsymbol{\phi}_t^\top \mathbf{P}_{t-1}\boldsymbol{\phi}_t,\ \varepsilon\right),$$

- a safety reset to $\mathbf{P}_t = q_0\mathbf{I}$ and $\mathbf{W}_t = \mathbf{0}$ whenever non-finite entries (NaN or Inf) are detected.

These mechanisms preserve the stability of the consequent layer and avoid catastrophic divergence when unexpected patterns appear in the stream.

1.5 Rule Pruning and Similarity Control

To keep the rule base compact and interpretable, ENFS-Uni0-Evolving applies two complementary mechanisms after each potential rule creation:

1. **Similarity-based pruning.** For any pair of rules (i, j), their similarity is defined as

$$\text{sim}(i,j) = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{c}_j\|^2}{\sigma_i^2 + \sigma_j^2 + \epsilon}\right). \tag{1.18}$$

   If the similarity between a newly created rule and an older rule exceeds a threshold $\theta \in (0, 1)$, the new rule is removed. This prevents redundant rules from cluttering the model and makes the set of extracted rules easier to interpret.

2. **Maximum-rule budget.** If the number of rules exceeds $R_{\max}$, the rule with the lowest support s_r (least used by the stream) is pruned. This enforces an upper bound on model complexity and encourages competition among rules, where only structurally meaningful ones survive.

1.6 Online Interpretability Over Data Streams

A central motivation for using *evolving* neuro-fuzzy models in software-engineering analytics is not merely to follow the temporal dynamics of software projects, but also to *understand* how these dynamics are reflected internally in the model structure. In this work, the ENFS-Uni0-Evolving model is instrumented with explicit interpretability metrics that evolve over time, so that the same data stream used for training also generates interpretable indicators of structural changes, attribute relevance, and potential concept drifts.

Let $t \in \{1, \ldots, T\}$ denote the time index (i.e., order of arrival of stream samples) and let $R(t)$ be the number of active rules after the update at time t. For each rule $r \in \{1, \ldots, R(t)\}$ the model maintains:

- a support count $s_r(t)$, representing how many samples have updated the rule;
- a vector of relevance weights $\mathbf{w}_r(t) = \big(w_{r,1}(t), \ldots, w_{r,d}(t)\big)$, updated incrementally following Lughofer's feature relevance mechanism;
- a consequent vector $\mathbf{v}_r(t)$, implicitly stored in the rows of the global RLS parameter matrix $\mathbf{W}_t$.

Below we describe the three major interpretability axes considered in this thesis: rule evolution, feature relevance evolution, and drift awareness.

1.6.1 Rule Evolution Curves

For every rule r, the temporal behaviour of its support is defined by

$$t \longmapsto s_r(t), \qquad t = 1, \ldots, T. \tag{1.19}$$

Stable increasing trajectories indicate persistent rules associated with regions frequently activated by the data stream; nearly flat trajectories indicate rules rarely used or possibly linked to transient patterns.

The global structural evolution is captured by the curve:

$$t \longmapsto R(t), \tag{1.20}$$

which reflects the model's structural complexity over time. Increments in $R(t)$ denote creation of new data clouds (new patterns), whereas decrements indicate merging or pruning of redundant rules.

In practice, rule-support curves and $R(t)$ are plotted together with the prequential accuracy, allowing visual correlation between structural dynamics and predictive performance.

1.6.2 Feature Relevance Evolution

For each rule r and each attribute j, the model maintains a time-varying relevance weight $w_{r,j}(t) \in [0, 1]$, updated via Lughofer's incremental feature relevance strategy:

$$w_{r,j}(t+1) = f\left(w_{r,j}(t), \text{ local contribution}_{r,j}(t+1)\right), \tag{1.21}$$

where the update function $f(\cdot)$ is monotonic with respect to the observed local discriminative contribution of attribute j, and includes temporal stabilization terms.

For visualization it is useful to apply a per-rule normalization:

$$\tilde{w}_{r,j}(t) = \frac{w_{r,j}(t)}{\sum_{\ell=1}^{d} w_{r,\ell}(t) + \varepsilon}, \quad \varepsilon > 0, \tag{1.22}$$

such that $\sum_j \tilde{w}_{r,j}(t) \approx 1$. Heatmaps of $\tilde{w}_{r,j}(t)$ reveal:

- globally important attributes (high and stable relevance);
- attributes that become relevant only during certain periods, potentially signalling regime changes in the development process;
- consistently irrelevant attributes, candidates for interpretable feature reduction.

1.6.3 Drift Awareness and Structural Markers

Software project metrics often exhibit *concept drift* due to changes in requirements, teams, technology stacks, release pressure, or workflow policies. Such drifts alter the relationship between input metrics (e.g., number of participants, issue churn, reviewer dynamics) and the target variable (e.g., delay vs. no delay).

In the ENFS-Uni0-Evolving model, drift awareness emerges from three parallel signals:

1. **Structural changes in the rule base**, captured by:

$$\Delta R(t) = R(t) - R(t-1), \tag{1.23}$$

   as well as sudden changes in the support of specific rules.

2. **Performance changes**, measured through prequential accuracy:

$$\mathrm{Acc}(t) = \frac{1}{t} \sum_{\tau=1}^{t} \mathbf{1}\{\hat{y}_\tau = y_\tau\}. \tag{1.24}$$

3. **Feature relevance reallocation**, detected when certain rule weights begin to dominate or fade, often reflecting meaningful process shifts.

Times where large structural changes occur (e.g., multiple new rules created in a short time window, or significant redistribution of relevance weights) can be marked on plots as potential drift points.

1.7 Fuzzy Rule Extraction and Linguistic Description

Although the ENFS-Uni0-Evolving model operates with numeric rule parameters, its purpose is to ultimately produce interpretable fuzzy rules. Each rule r is defined by its centre $\mathbf{c}_r \in \mathbb{R}^d$, its radius $\sigma_r > 0$, its relevance weights $\mathbf{w}_r(t)$, and its consequent vector $\mathbf{v}_r(t)$ (stored in the global RLS matrix).

1.7.1 Gaussian Fuzzification

For each input dimension j and rule r, the Gaussian membership degree is:

$$\mu_{r,j}(x_j) = \exp\left(-\frac{(x_j - c_{r,j})^2}{2\sigma_r^2}\right), \tag{1.25}$$

yielding the vector of local memberships $\boldsymbol{\mu}_r(\mathbf{x}_t)$ for a stream sample $\mathbf{x}_t$.

1.7.2 OR-Neuron Aggregation

Instead of the original UniNull neuron, this version employs a more stable probabilistic OR-aggregation:

$$y_r(\mathbf{x}_t) = \frac{\sum_{j=1}^{d} w_{r,j}(t)\, \mu_{r,j}(x_{t,j})}{\sum_{j=1}^{d} w_{r,j}(t) + \varepsilon}, \quad \varepsilon > 0. \tag{1.26}$$

Thus, the rule activates strongly whenever *at least one* relevant feature matches the local data cloud.

The RLS layer receives the activation vector:

$$\boldsymbol{\phi}_t = [1,\ y_1(\mathbf{x}_t), \ldots, y_{R(t)}(\mathbf{x}_t)]^\top, \tag{1.27}$$

and learns a globally linear mapping from rule activations to class scores.

1.7.3 Linguistic Rule Templates

Rule r can be translated into a linguistic statement of the form:

$$\begin{aligned} &\text{IF } (x_1 \text{ is } A_{r,1} \text{ with impact } w_{r,1}(t)) \ \text{ OR } \cdots \ \text{ OR} \\ &(x_d \text{ is } A_{r,d} \text{ with impact } w_{r,d}(t)) \ \text{ THEN class } = \arg\max_c v_{r,c}(t), \end{aligned} \tag{1.28}$$

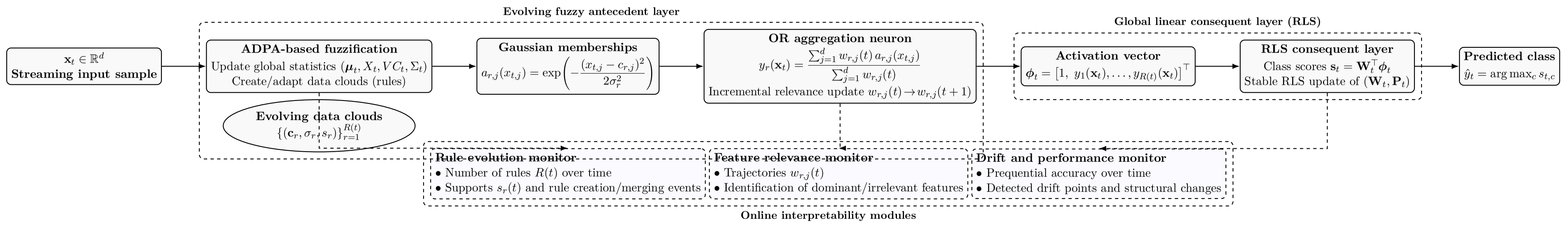
where the linguistic terms $A_{r,j}$ come from thresholding the Gaussian membership:

$$\mu_{r,j}(x_j) \mapsto \begin{cases} \text{“low”} & \mu_{r,j}(x_j) < \tau_1, \\ \text{“medium”} & \tau_1 \le \mu_{r,j}(x_j) < \tau_2, \\ \text{“high”} & \mu_{r,j}(x_j) \ge \tau_2, \end{cases} \qquad 0 < \tau_1 < \tau_2 < 1. \tag{1.29}$$

Because impact weights $w_{r,j}(t)$ are updated incrementally, rule (1.28) directly shows which attributes are truly responsible for the rule's decision at different stages of the project.

Conclusion

This chapter provided a mathematically rigorous and implementation-faithful description of the ENFS-Uni0-Evolving classifier used in this dissertation. The model extends the original ENFS-Uni0 in several ways, including stability enhancements, feature relevance tracking, and explicit interpretability for streaming data.



$\mathbf{x}_t \in \mathbb{R}^d$
Streaming input sample
Evolving fuzzy antecedent layer
ADPA-based fuzzification
Update global statistics $(\boldsymbol{\mu}_t, X_t, VC_t, \Sigma_t)$
Create/adapt data clouds (rules)
Evolving data clouds
$\{(\mathbf{c}_r, \sigma_r, s_r)\}_{r=1}^{R(t)}$
Gaussian memberships
$a_{r,j}(x_{t,j}) = \exp\left(-\frac{(x_{t,j}-c_{r,j})^2}{2\sigma_r^2}\right)$
OR aggregation neuron
$y_r(\mathbf{x}_t) = \frac{\sum_{j=1}^d w_{r,j}(t)\, a_{r,j}(x_{t,j})}{\sum_{j=1}^d w_{r,j}(t)}$
Incremental relevance update $w_{r,j}(t) \to w_{r,j}(t+1)$
Global linear consequent layer (RLS)
Activation vector
$\boldsymbol{\phi}_t = [1,\ y_1(\mathbf{x}_t), \ldots, y_{R(t)}(\mathbf{x}_t)]^\top$
RLS consequent layer
Class scores $\mathbf{s}_t = \mathbf{W}_t^\top \boldsymbol{\phi}_t$
Stable RLS update of $(\mathbf{W}_t, \mathbf{P}_t)$
Predicted class
$\hat{y}_t = \arg\max_c s_{t,c}$
Rule evolution monitor
• Number of rules $R(t)$ over time
• Supports $s_r(t)$ and rule creation/merging events
Feature relevance monitor
• Trajectories $w_{r,j}(t)$
• Identification of dominant/irrelevant features
Drift and performance monitor
• Prequential accuracy over time
• Detected drift points and structural changes
Online interpretability modules

Figure 1.1: High-level architecture of the ENFS-Uni0-Evolving model, highlighting the ADPA-based fuzzy antecedent layer, the OR aggregation neuron with incremental feature relevance, the RLS consequent layer, and the online interpretability modules.

Bibliography

[1] de Campos Souza, P. V. and Lughofer, E. (2021). An evolving neuro-fuzzy system based on uni-nullneurons with advanced interpretability capabilities. *Neurocomputing*, 451:231–251.