Kangin, D., Angelov, P., Iglesias, J.A. (2016). Autonomously evolving classifier TEDAClass. *Information Sciences*, 366, pp. 1-11.

DOI: 10.1016/j.ins.2016.05.012

Autonomously Evolving Classifier TEDAClass

Dmitry Kangin[a], Plamen Angelov[a], José Antonio Iglesias[b]

[a] *Data Science Group, Computing & Communications, Lancaster University, UK {d.kangin, p.angelov}@lancaster.ac.uk*
[b] *Carlos III University of Madrid, Spain jiglesia@inf.uc3m.es*

Abstract

In this paper we introduce a classifier named TEDAClass (Typicality and Eccentricity based Data Analytics Classifier) which is based on the recently proposed AnYa type fuzzy rule based system. Specifically, the rules of the proposed classifier are defined according to the recently proposed TEDA framework. This novel and efficient systematic methodology for data analysis is a promising addition to the traditional probability as well as to the fuzzy logic. It is centred at non-parametric density estimation derived from the data sample. In addition, the proposed framework is computationally cheap and provides fast and exact per-point processing of the data set/stream. The algorithm is demonstrated to be suitable for different classification tasks. Throughout the paper we give evidence of its applicability to a wide range of practical problems. Furthermore, the algorithm can be easily adapted to different classical data analytics problems, such as clustering, regression, prediction, and outlier detection. Finally, it is very important to remark that the proposed algorithm can work "from scratch" and evolve its structure during the learning process.

Keywords: Classifiers, evolving systems, TEDA, fuzzy systems

1. Introduction

Nowadays, the classical problems of machine learning are widely studied. However, many architecturally different algorithms still suffer from a number of limitations. If we take into account how they work, we can consider three different kinds of algorithms: offline, incremental and online learning. The offline algorithms are trained once, and then do not admit any modifications to the classifier parameters or structure. Hence, the performance of these algorithms is only restricted to the patterns observed during the training stage. Furthermore, the incremental algorithms admit modifications of (some of) the parameters, but they generally demand previous data samples (all or some part) to be stored in the memory. They do not allow modification of the classifier structure, and very often they are not sufficient for practical needs. Finally, the algorithms, which implement online learning, do not need to memorise all previous training samples. This kind of algorithms are usually computationally efficient, recursive procedure, and applicable for online real-time applications.

However, none of these three types of algorithms involve any kind of adaptation of the structure to take into account the newly emerging or dynamically evolving data

patterns which were not present in the initial training data set/stream. For this reason, a recently emerged branch of machine learning aiming to address such problems: Evolving Intelligent Systems (EIS) [1]. The algorithm developed in this work is based on the paradigm of such systems that is a distinguishable feature of the proposed algorithm amongst the well-studied techniques. One of the most renowned machine learning challenges is the classification problem. Some of the versions of the problem statement are domain-specific, while others are defined generally, and then adapted for a particular problem. Hereafter we provide a classifier which does not have any user- or problem-specific parameter.

One of the most renowned machine learning challenges is the classification problem. Some of the versions of the problem statement are domain-specific, while others are defined generally, and then adapted for a particular problem. Hereafter we provide a classifier which does not have any user- or problem-specific parameter.

Define some object set Ω, and a finite space of object classes, referred as C. Then define a subset, $\Omega_L \subset \Omega$ called learning set, for each of the following functions defined as: $F_L : \Omega_L \rightarrow C$. The problem is to build a function $F : \Omega \rightarrow C$, approximating the function F_L on the set Ω_L. The assumption is that F will be a good mapping for further objects, Ω_V where the index V denotes validation. $\Omega_V \cap \Omega_L = \emptyset, \Omega_V, \Omega_L \subset \Omega$.

In this paper, we address the problem of a general purpose classifier, which needs to meet the following requirements:

- ***Ability of incremental learning:*** the classifier should be one-pass. Thus, each sample is proceeded once and there is no need to re-learn any previous samples once the new training data appear.

- ***Online design:*** the classifier should proceed the samples at the time of their arrival, the algorithm should not rely on memorised context of all data samples. There is no need to store all the given samples, but only some descriptor data of them, ideally with fixed size upper bound.

- ***Evolving structure:*** the classifier should be able to start learning "from scratch" and its parameters and structure should be self-adapting on-line. In addition, it is non-parametric.

- ***Speed and memory efficiency:*** The algorithm should be computationally effective so it can be applied to online real-time (or near real-time) applications.

In order to meet all these requirements, we propose a novel and efficient classifier named TEDAClass in which the novel TEDA framework [2] is used within a system of evolving fuzzy AnYa-type rules [3]. This combination enables to build easily interpretable model for classification. The most important contribution of this paper is the novel classifier TEDAClass. In addition, we also present the derivations of incremental calculation formulae, which are necessary for updating the incremental fuzzy rules of the classifier. The novel classifier has been successfully tested on several benchmark problems.

The reminder of the paper is organised as follows. Section II describes the related works. Section III gives a brief description of the previously proposed TEDA

approach. Section IV contains necessary novel derivations of the incremental formulae within TEDA framework, necessary for further algorithm description, and sections V formulates the classifier itself. The experimental data is presented in section VI. Finally, Section VII contains future work and concluding remarks.

2. Related work

There are many different well-studied algorithms of classification. The most relevant algorithms include Support Vector Machines (SVMs) [4], Neural Networks (Single- and Multi-layer neural networks [5], [6], Radial Basis Function networks [7]), Bayesian networks [8], fuzzy rule-based models [9] or decision trees [10]. But from a wide variety of classification methods, we particularly pay attention to fuzzy logic methods, which offer well-developed apparatus for evolving system development [11]. This kind of methods gives an essential contribution to this article. Here we review the state-of-the art algorithms in the prism of evolving systems development.

The evolving systems, based on fuzzy logic approach, use a complex and varying combination of simple models which are adjustable "from scratch", and capable of changing its own structure during the system learning [12]. Amongst those algorithms are recently introduced *eClass* [9], *AutoClass* [13], DEC [14], FLEXFISClass [15], or self-evolving classifier, described in [16].

SVMs [4] provide rigorous and sparse solution for various classification problems, and are especially well renowned for the results obtained via the so-called kernel trick. They can also be implemented incrementally. If we discard some of the support vectors according to some rule, it is possible to obtain Evolving Systems based on SVM.

Evolving Spiking Neural Networks [17] is one of the evolving approaches based on Artificial Neural Networks. However, reasonable efforts should be usually undertaken to learn it properly without falling into a local extremum (for example, in case of [17], the genetic algorithms are utilised for the optimisation).

Nowadays, fuzzy rule-based evolving systems are predominantly of the type of Takagi-Sugeno [18] or Mamdani [19]:

Mamdani:

$$\text{IF } (\text{ant}_i(\vec{x})) \text{ THEN } (y_i \text{ IS } Y_i), \tag{1}$$

Takagi-Sugeno:

$$\text{IF } (\text{ant}_i(\vec{x})) \text{ THEN } (y_i = \vec{x}^T \Theta_i), \tag{2}$$

where y_i is an outcome of the i-th rule, $i \in [1 \dots N]$, Θ_i is a design matrix for linear regression, $\vec{x}$ is an input data vector, $\text{ant}_i(\vec{x})$ is the antecedent of the fuzzy rule.

In both cases, the antecedent is expressed as

$$\text{ant}_i(\vec{x}) : x^1 \text{ IS } L_i^1 \text{ AND } x^2 \text{ IS } L_i^2 \text{ AND } \cdots \text{ AND } x^n \text{ IS } L_i^n, \tag{3}$$

where n is a dimensionality of the vector $\vec{x} = (x^1, x^2, \dots, x^n)$, and $\vec{L}_i = (L_i^1, L_i^2, \dots, L_i^n)$ is a reference vector for the fuzzy rule.

In the approach that we present in this paper, we use a more general approach which is called AnYa [3]:

AnYa:

$$\text{IF } (\vec{x} \text{ IS LIKE } \vec{L_i}) \text{ THEN } (y_i = \vec{x}^T \Theta_i), \tag{4}$$

This approach gives more compact, non-parametric form and does not require to consider each of the dimensions separately and pre-defining a membership function or pdf for each one of them.

The classification approach proposed in this paper is based on recently described TEDA framework [2]. In this classifier, we combine TEDA with fuzzy logic approach, which is a computationally cheap alternative to the traditional frequentistic probability theory and gives a systematic methodology. In addition, the proposed classifier gives a simple and transparent interpretation of the properties of the data sets. Finally, it can be easily combined with fuzzy rules, as it gives easy interpretable statistical metrics for data.

3. A general introduction to TEDA approach

In this section, we briefly describe the TEDA framework [2], which gives a new systematic approach to a "per point" online data analysis without making unrealistic assumptions.

Let us have some feature space $\mathfrak{X} \subseteq \mathbb{R}^n$, which we consider to be a space of data samples, or observations. For this space, we can define some distance $d(\vec{x}, \vec{y})$ (i.e. Euclidean, Mahalanobis[21], L^1, cosine or any other convenient for the case) between feature space elements $\vec{x}, \vec{y}$.

Then, let us consider the data samples as an ordered sequence

$$\{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_k, \dots\}, \vec{x}_i \in \mathbb{R}^n, i \in \mathbb{N}. \tag{5}$$

where index k's physical meaning, typically, is the instant of time when the data sample has arrived. For the sake of simplicity, index k will be referred to as time instant further.

For the whole data set/stream, we can define sum distance to some particular point $\vec{x} \in \mathfrak{X}$ for each element up to the k-th one:

$$\pi^k(\vec{x}) = \sum_{i=1}^{k} d(\vec{x}, \vec{x}_i). \tag{6}$$

The eccentricity at the time instant k can be defined as [2]

$$\xi^k(\vec{x}) = \frac{\sum_{i=1}^{k} d(\vec{x}, \vec{x}_j)}{\sum_{i=1}^{k} \sum_{j=i+1}^{k} d(\vec{x}_i, \vec{x}_j)} = 2\frac{\sum_{i=1}^{k} d(\vec{x}, \vec{x}_j)}{\sum_{i=1}^{k} \sum_{j=1}^{k} d(\vec{x}_i, \vec{x}_j)} = \frac{2\pi^k(\vec{x})}{\sum_{i=1}^{k} \pi^k(\vec{x_i})}. \tag{7}$$

Here $k \geq 2, \sum_{i=1}^{k} \pi^k(\vec{x_i}) > 0$.

As a complement to the eccentricity, the typicality is also defined as [1]:

$$\tau^k(\vec{x}) = 1 - \xi^k(\vec{x}). \tag{8}$$

The eccentricity and typicality are both bounded [2]:

$$0 \leq \xi^k(\vec{x}) \leq 1, \sum_{i=1}^{k} \xi^k(\vec{x}_i) = 2, \tag{9}$$

$$0 \leq \tau^k(\vec{x}) \leq 1, \sum_{i=1}^{k} \tau^k(\vec{x}_i) = k-2, k \geq 2, \sum_{i=1}^{k} \pi^k(\vec{x_i}) > 0. \tag{10}$$

Normalised eccentricity and typicality can also be defined as [1]:

$$0 \leq \zeta^k(\vec{x}) \leq 1, \sum_{i=1}^{k} \zeta^k(\vec{x}_i) = 1, k \geq 2, \tag{11}$$

$$t^k(\vec{x}) = \frac{\tau^k(\vec{x})}{k-2}, \sum_{i=1}^{k} \tau^k(\vec{x}_i) = 1, k > 2, \tag{12}$$

Typicality and eccentricity can be easily calculated recursively that gives significant benefits for online data processing. They are exact and do not require the very restrictive assumptions which are made for the traditional probability theory, yet they have very similar properties to it as well as to the fuzzy set theory. We concentrate on Euclidean and Mahalanobis [20] distance especially, but it does not impact the generalisation of the method to other metrics. The calculations using these distances are exact and do not require the very restrictive assumptions which are made for the traditional probability theory. In addition, they have very similar properties as well as to the fuzzy sets theory. In the next two sections we will describe how our proposed TEDAClass algorithm works taking into account these two distances.

4. TEDA calculations

In this section, we describe in the detail the process to calculate the eccentricity and typicality recursively using Euclidean and Mahalanobis distances (subsections 4.1 and 4.2). It is important to remark that the values obtained are exact (not approximated or learned) and restrictive assumptions are not required. In the case of Mahalanobis distance, we also describe in detail how to calculate it recursively via recursive covariance matrix and mean calculation (subsection 4.3). Since all these values are calculated recursively, TEDA can be used in an online manner, what is essential to be used in many real environment.

4.1. TEDA calculation: Euclidean distance case

Let us assume that the distance is Euclidean, and then derive some formulae describing the process of recursive calculation of the eccentricity and typicality [21]:

$$\xi^k(\vec{x}) = 2\frac{\sum_{i=1}^{k} d(\vec{x},\vec{x}_j)}{\sum_{i=1}^{k}\sum_{j=1}^{k} d(\vec{x}_i,\vec{x}_j)} = 2\frac{\sum_{i=1}^{k}(\vec{x}_i-\vec{x})^T(\vec{x}_i-\vec{x})}{\sum_{i=1}^{k}\sum_{j=1}^{k}(\vec{x}_i-\vec{x}_j)^T(\vec{x}_i-\vec{x}_j)} =$$

$$= \left\{ \vec{\mu}_x^k = \sum_{i=1}^{k}\frac{\vec{x}_i}{k}, \mu_{\vec{x}^T\vec{x}}^k = \sum_{i=1}^{k}\vec{x}_i^T\vec{x}_i, \{\sigma_x^k\}^2 = \sum_{i=1}^{k}\frac{(\vec{x}_i-\vec{\mu}_x^k)^T(\vec{x}_i-\vec{\mu}_x^k)}{k} \right\} =$$

$$= \frac{2k\left(\mu_{\vec{x}^T\vec{x}}^k - 2[\vec{\mu}_x^k]^T\vec{x} + \vec{x}^T\vec{x}\right)}{k^2(2\mu_{\vec{x}^T\vec{x}}^k - 2[\vec{\mu}_x^k]^T\vec{\mu}_x^k)} = \{\textbf{using variance properties}\} =$$

$$= \frac{2k\left(\mu_{\vec{x}^T\vec{x}}^k - 2[\vec{\mu}_x^k]^T\vec{x} + \vec{x}^T\vec{x}\right)}{2k^2[\sigma_x^k]^2} = \frac{[\sigma_x^k]^2 + [\vec{\mu}_x^k]^T\vec{\mu}_x^k - 2[\vec{\mu}_x^k]^T\vec{x} + \vec{x}^T\vec{x}}{k[\sigma_x^k]^2} =$$

$$= \frac{[\sigma_x^k]^2 + (\vec{\mu}_x^k-\vec{x})^T(\vec{\mu}_x^k-\vec{x})}{k[\sigma_x^k]^2}. \tag{13}$$

In summary,

$$\xi^k(\vec{x}) = \frac{1}{k} + \frac{(\vec{\mu}_x^k-\vec{x})^T(\vec{\mu}_x^k-\vec{x})}{k[\sigma_x^k]^2}. \tag{14}$$

Mean $\vec{\mu}_x^k$ and variance σ_x^k can be calculated recursively:

$$\vec{\mu}_x^k = \frac{k-1}{k}\vec{\mu}_x^{k-1} + \frac{\vec{x}^k}{k}, k \geq 1, \vec{\mu}_x^0 = \vec{0}, \tag{15}$$

$$\mu_{\vec{x}^T\vec{x}}^k = \frac{k-1}{k}\mu_{\vec{x}^T\vec{x}}^{k-1} + \frac{[\vec{x}^k]^T\vec{x}^k}{k}, k \geq 1, \mu_{\vec{x}^T\vec{x}}^0 = 0, \tag{16}$$

$$[\sigma_x^k]^2 = \mu_{\vec{x}^T\vec{x}}^k - [\vec{\mu}_x^k]^T\vec{\mu}_x^k. \tag{17}$$

The typicality, similarly, can be represented as

$$\tau^k(\vec{x}) = 1 - \xi^k(\vec{x}) = \frac{k-1}{k} - \frac{(\vec{\mu}_x^k-\vec{x})^T(\vec{\mu}_x^k-\vec{x})}{k[\sigma_x^k]^2}. \tag{18}$$

Then one can give an interpretation of normalised typicality and eccentricity and prove that the well-known Chebyshev inequality can be derived as a special case of TEDA if used for anomaly detection [21].

One can obtain the following inequalities for typicality:

$$t^k(\vec{x}) = \frac{1}{k-2} - \frac{1}{k(k-2)} - \frac{(\vec{\mu}_x^k-\vec{x})^T(\vec{\mu}_x^k-\vec{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \tag{19}$$

$$\frac{k-1}{k(k-2)} - \frac{(\vec{\mu}_x^k-\vec{x})^T(\vec{\mu}_x^k-\vec{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \tag{20}$$

$$\frac{1}{k} + \frac{1}{k(k-2)} - \frac{(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x})}{k(k-2)[\sigma_x^k]^2} > \frac{1}{k}, \tag{21}$$

$$\frac{1}{k(k-2)} > \frac{(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x})}{k(k-2)[\sigma_x^k]^2}, \tag{22}$$

$$(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x}) < [\sigma_x^k]^2, \tag{23}$$

The typicality can be interpreted in the following way. We see that supposing normalised typicality be greater than $1/k$ the results are shown to be exactly the same as those of the Chebyshev inequality for '$m\sigma$' rule. Moreover, it is easy to check, that for '$m\sigma$' rule we have:

$$t^k(\vec{x}) = \frac{1}{k-2} - \frac{1}{k(k-2)} - \frac{(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x})}{k(k-2)[\sigma_x^k]^2} > \frac{-n^2 + k - 1}{k(k-2)}. \tag{24}$$

Then, in order to normalise eccentricity we can check the similar equations, leading us also to Chebyshev inequality:

$$(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x}) < m^2[\sigma_x^k]^2, \tag{25}$$

$$\frac{(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x})}{2k[\sigma_x^k]^2} < \frac{m^2}{2k}, \tag{26}$$

$$\zeta^k(\vec{x}) = \frac{1}{2k} + \frac{(\vec{\mu}_x^k - \vec{x})^T(\vec{\mu}_x^k - \vec{x})}{2k[\sigma_x^k]^2} < \frac{m^2 + 1}{2k}, \tag{27}$$

4.2. TEDA calculation: Mahalanobis distance

For cases involving high dimensional data sets (e.g., image feature mapping), the '3σ' rule for the Euclidean distance can result in a low accuracy. In these cases, a more sophisticated dependencies should be used, enabling control over the spread of the fuzzy rules. Similarly to the case with using Euclidean distance, we can use Mahalanobis distance [20] for TEDA calculation. Then we have:

$$\xi^k(\vec{x}) = \frac{2\sum_{i=1}^k d(\vec{x}_i, \vec{x})}{\sum_{i=1}^k \sum_{j=1}^k d(\vec{x}_i, \vec{x}_j)} = \frac{2\sum_{i=1}^k (\vec{x} - \vec{x}_i)^T[\Sigma_x^k]^{-1}(\vec{x} - \vec{x}_i)}{\sum_{i=1}^k \sum_{j=1}^k (\vec{x}_j - \vec{x}_i)^T[\Sigma_x^k]^{-1}(\vec{x}_j - \vec{x}_i)}, \tag{28}$$

where covariance matrix is defined as follows:

$$\Sigma_x^k = \frac{1}{k}\sum_{m=1}^k (\vec{x}_m - \vec{\mu}_x^k)(\vec{x}_m - \vec{\mu}_x^k)^T. \tag{29}$$

Let us then consider the denominator and simplify it further:

$$\sum_{i=1}^k \sum_{j=1}^k d(\vec{x}_i, \vec{x}_j) = \sum_{i=1}^k \sum_{j=1}^k (\vec{x}_j - \vec{x}_i)^T[\Sigma_x^k]^{-1}(\vec{x}_j - \vec{x}_i). \tag{30}$$

$$\sum_{i=1}^{k}\sum_{j=1}^{k} d(\vec{x}_i,\vec{x}_j) = 2\sum_{i=1}^{k}\sum_{j=1}^{k} \vec{x}_i^T[\Sigma_x^k]^{-1}\vec{x}_i - 2\sum_{i=1}^{k}\sum_{j=1}^{k} \vec{x}_i^T[\Sigma_x^k]^{-1}\vec{x}_j =$$

$$= 2k\sum_{i=1}^{k} \vec{x}_i^T[\Sigma_x^k]^{-1}\vec{x}_i - 2k^2[\vec{\mu}_x^k]^T[\Sigma_x^k]^{-1}\vec{\mu}_x^k =$$

$$= 2k\sum_{i=1}^{k} \vec{x}_i^T\left[\frac{1}{k}\sum_{m=1}^{k}(\vec{x}_m-\vec{\mu}_x^k)(\vec{x}_m-\vec{\mu}_x^k)^T\right]^{-1}\vec{x}_i -$$

$$-2k^2[\vec{\mu}_x^k]^T\left[\frac{1}{k}\sum_{m=1}^{k}(\vec{x}_m-\vec{\mu}_x^k)(\vec{x}_m-\vec{\mu}_x^k)^T\right]^{-1}\vec{\mu}_x^k =$$

$$= \sum\left[2k\left(\sum_{i=1}^{k}\vec{x}_i\vec{x}_i^T\right)\otimes\left(\frac{1}{k}\sum_{j=1}^{k}\vec{x}_j\vec{x}_j^T - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right)^{-1}\right] -$$

$$-\sum\left[2k^2\left(\vec{\mu}_x^k[\vec{\mu}_x^k]^T\right)\otimes\left(\frac{1}{k}\sum_{j=1}^{k}\vec{x}_j\vec{x}_j^T - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right)^{-1}\right] = \tag{31}$$

$$= \left\{\mu_{\vec{x}\vec{x}^T} = \frac{1}{k}\sum_{j=1}^{k}\vec{x}_j\vec{x}_j^T\right\} =$$

$$= 2k^2\sum\left(\left\{\vec{\mu}_{\vec{x}\vec{x}^T} - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right\}\otimes\left\{\vec{\mu}_{\vec{x}\vec{x}^T} - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right\}^{-1}\right) =$$

$$= \textbf{thanks to the covariance matrix simmetricity} =$$

$$= 2k^2\sum\left(\left\{\mu_{\vec{x}\vec{x}^T} - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right\}\left\{\mu_{\vec{x}\vec{x}^T} - \vec{\mu}_x^k[\vec{\mu}_x^k]^T\right\}^{-1}\right)\otimes I_{n\times n} =$$

$$= 2k^2\mathbf{tr}(I_{n\times n}) = 2k^2 n.$$

Here $\otimes$ is an element-wise matrix multiplication operator, and $\sum$ denotes the sum of all the matrix elements.

Thus one can see that the denominator can be reduced to a constant. It gives us a reason to interpret typicality as a quantity proportional to Mahalanobis distance, and express typicality and eccentricity concepts in terms of Mahalanobis distance.

Then we can obtain:

$$\xi_k(\vec{x}) = \frac{\sum_{i=1}^{k}(\vec{x}-\vec{x}_i)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{x}_i)}{k^2 n} =$$

$$= \frac{\vec{x}^T \left[\Sigma_x^k\right]^{-1}\vec{x} - 2\left[\vec{\mu}_x^k\right]^T \left[\Sigma_x^k\right]^{-1}\vec{x} + \left[\vec{\mu}_x^k\right]^T \left[\Sigma_x^k\right]^{-1}\vec{\mu}_x^k}{kn} +$$

$$+ \frac{\sum_{i=1}^{k}\vec{x}_i^T \left[\Sigma_x^k\right]^{-1}\vec{x}_i - k[\vec{\mu}_x^k]^T \left[\Sigma_x^k\right]^{-1}\vec{\mu}_x^k}{k^2 n} = \tag{32}$$

$$= \frac{\vec{x}^T \left[\Sigma_x^k\right]^{-1}\vec{x} - 2\left[\vec{\mu}_x^k\right]^T \left[\Sigma_x^k\right]^{-1}\vec{x} + [\vec{\mu}_x^k]^T \left[\Sigma_x^k\right]^{-1}\vec{\mu}_x^k}{kn} + \frac{1}{k} =$$

$$= \frac{(\vec{x}-\vec{\mu}_x^k)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{\mu}_x^k)}{kn} + \frac{1}{k}.$$

Finally we can state the equations for Chebyshev inequality:

$$(\vec{x}-\vec{\mu}_x^k)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{\mu}_x^k) < m^2, \tag{33}$$

where m is a multi-dimensional threshold.

Then

$$\zeta^k(\vec{x}) = \frac{(\vec{x}-\vec{\mu}_x^k)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{\mu}_x^k)}{2kn} + \frac{1}{2k} < \frac{m^2+n}{2kn} \tag{34}$$

or

$$\zeta^k(\vec{x}) < \frac{m^2+n}{2kn} \tag{35}$$

Similarly for typicality we have:

$$t^k(\vec{x}) = \frac{1}{k-1}\left(1 - \frac{(\vec{x}-\vec{\mu}_x^k)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{\mu}_x^k)}{kn} - \frac{1}{kn}\right) > \frac{(k-1)n-m^2}{kn(k-1)}. \tag{36}$$

Therefore, we can specify a threshold for Mahalanobis distance, as well as calculate it recursively via recursive covariance matrix and mean calculation.

4.3. Recursive Covariance matrix Update

In this section, we describe other important contribution of this paper: how the recursive covariance matrix is updated in order to calculate Mahalanobis distance recursively:

$$d^k(\vec{x}) = (\vec{x}-\vec{\mu}_x^k)^T \left[\Sigma_x^k\right]^{-1}(\vec{x}-\vec{\mu}_x^k). \tag{37}$$

We should notice here, that the distance is scalar, whilst inverse covariance matrix $\left[\Sigma_x^k\right]^{-1}$ is a square matrix $n \times n$, where n is the dimensionality of the feature space.

We can easily estimate the mean value:

$$\vec{\mu}_x^k = \frac{k-1}{k}\vec{\mu}_x^{k-1} + \frac{\vec{x}_k}{k}, \vec{\mu}_x^0 = \vec{0}. \tag{38}$$

but we also need a recursive covariance matrix calculation. In what follows, we present the derivation of the method.

First, we express covariance matrix:

$$\Sigma_x^k = \frac{1}{k}\sum_{i=1}^{k}(\vec{x}_i - \vec{\mu}_i^k)(\vec{x}_i - \vec{\mu}_i^k)^T = \frac{1}{k}\sum_{i=1}^{k}\vec{x}_i\vec{x}_i^T - \vec{\mu}_x^k\left[\vec{\mu}_x^k\right]. \tag{39}$$

Then the recursive calculation may be applied:

$$\mu_{\vec{x}\vec{x}^T}^k = \frac{k-1}{k}\mu_{\vec{x}\vec{x}^T}^{k-1} + \frac{1}{k}\sum_{j=1}^{k}\vec{x}_j\vec{x}_j^T. \tag{40}$$

$$\mu_{\vec{x}^T\vec{x}}^k = \frac{k-1}{k}\mu_{\vec{x}^T\vec{x}}^{k-1} + \frac{1}{k}\sum_{j=1}^{k}\vec{x}_j^T\vec{x}_j. \tag{41}$$

$$\Sigma_x^k = \frac{k-1}{k}\Sigma_x^{k-1} + \frac{k^2-1}{k^2}\vec{x}_k\vec{x}_k^T + \frac{k-1}{k^2}\vec{\mu}_x^{k-1}(\vec{\mu}_x^{k-1} - 2\vec{x}_k)^T. \tag{42}$$

Thus we have derived a formula which can be used to update the covariance matrix. Then, we should derive the formulae for inverse matrix as well. Here we need to use the well-known Woodbury formula [22]:

$$(A+UBV)^{-1} = A^{-1} - A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1}. \tag{43}$$

We can use this formula twice, for the first and the second terms of the recursive expression, and then for the third one.

First we calculate p^{-1} for

$$p = \frac{k-1}{k}\Sigma_x^{k-1} + \frac{k^2-1}{k^2}\vec{x}_k\vec{x}_k^T. \tag{44}$$

Here we substitute $\frac{k}{k-1}[\Sigma_x^{k-1}]^{-1}$ with A^{-1}, $B=1$, $U=\frac{k^2-1}{k^2}\vec{x}_k$, $V=x_{k+1}^T$. Then:

$$p^{-1} = \frac{k}{k-1}[\Sigma_x^{k-1}]^{-1} +$$

$$+\frac{k}{k-1}[\Sigma_x^{k-1}]^{-1}\frac{k+1}{k}\vec{x}_k(1 + \frac{k+1}{k}\vec{x}_k^T[\Sigma_x^{k-1}]^{-1}x_k)^{-1}\times$$

$$\times x_k^T[\Sigma_x^{k-1}]^{-1} = \frac{k}{k-1}\left[\Sigma_x^{k-1}\right]^{-1} + \tag{45}$$

$$+\frac{\frac{k}{k-1}\left[\Sigma_x^{k-1}\right]^{-1}\frac{k+1}{k}\vec{x}_k\vec{x}_k^T\left[\Sigma_x^{k-1}\right]^{-1}}{1+\frac{k+1}{k}\vec{x}_k^T\left[\Sigma_x^{k-1}\right]^{-1}\vec{x}_k},$$

$$\Sigma_x^k = p + \frac{1}{k^2}\vec{\mu}_x^{k-1}(\vec{\mu}_x^{k-1} - 2\vec{x}_k)^T.$$

Here we substitute p^{-1} with A^{-1}, $B=1$, $U=\frac{k-1}{k^2}\mu_k^{k-1}$, $V=(\mu_x^{k+1}-2\vec{x}_k)^T$ and finally we obtain:

$$\begin{gathered}\left[\Sigma_x^k\right]^{-1}=p^{-1}+\\ +p^{-1}\frac{(k-1)\vec{\mu}_x^k)}{k^2}(1+\frac{k-1}{k^2}(\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}\vec{\mu}_x^k)^(-1)\times\\ \times(\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}=p^{-1}+\\ +\frac{p^{-1}\frac{(k-1)\vec{\mu}_x^{k\ 1}}{k^2}(\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}}{1+\frac{(k-1))}{k^2}(\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}\vec{\mu}_x^{k-1}}\end{gathered} \tag{46}$$

Finally, we get:

$$\begin{gathered}\left[\Sigma_x^k\right]^{-1}=p^{-1}+\\ +\frac{p^{-1}\frac{(k-1)\vec{\mu}_x^{k\ 1}}{k^2}\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}}{1+\frac{(k-1)}{k^2}(\vec{\mu}_x^{k-1}-2\vec{x}_k)^Tp^{-1}\vec{\mu}_x^{k-1}}\end{gathered} \tag{47}$$

5. Our Proposed TEDA Classifier

In this section we explain in detail the classifier based on the TEDA framework that we propose as main contribution in this paper. This proposed classifier is presented in its two variations: based on Euclidean and Mahalanobis distances (which have been previously detailed).

As we have already explained in section II, our proposed classifier is based on the general approach called AnYa [3], which is compact and non-parametric. If we describe the classifier in terms of evolving AnYa system, we obtain:

$$F=\{R_i\}, i=1,\vec{N}, R_i(\vec{x}): \text{IF}(\vec{x}\sim\vec{x_i^*})\ \text{THEN}\ y_i=\vec{x}^T\Theta_i. \tag{48}$$

where F is a fuzzy rule set, each defined over one data cloud, R_i is a particular rule of this set, $y_i\in C, \vec{x}\in\Omega, F:\Omega\to C, \Omega$ is a feature space, C is a class space, $\vec{x_i^*}$ is cloud representative point (focal point), Θ_i is a design matrix for linear regression, $\sim$ is some relation of closeness (association/membership) of the point $\vec{x}$ to the fuzzy rule $R_i(\vec{x})$ corresponding to the data cloud. The data cloud is like a cluster, in the sense it is a group of similar and close data points, but it differs from clusters in sense that it has not any boundaries or specified shapes [22]. For this reason, an important aspect to consider in this approach is that the clouds are described by a set of data samples but they do not have specific shape.

The set of fuzzy rules can change with each sample both quantitatively (changing focal points and design matrices) and qualitatively (fuzzy rule set cardinality fluctuations) [23]. This aspect is essential since it ensures the evolving structure of the classifier.

The procedure of classifier learning for TEDAClass highly resembles the procedure used by *eClass* [10] and *AutoClass* [13]. However, the way TEDAClass determines the closeness to the specified data cloud, as we explained below, has significantly changed. In TEDAClass, we use typicality and eccentricity concepts to calculate closeness to the rule. The more is the value of typicality, the closer is the object to the rule. In addition, for each new sample of training data, we recalculate typicality of the point. The firing strength of each rule is determined as follows:

$$w_i^k(\vec{x}) = \frac{t_i^k(\vec{x})}{\Sigma_{j=1}^N t_j^k(\vec{x})}, \tag{49}$$

where normalised typicality t_i^k is given over all the elements of the fuzzy rule. We assign each new coming training point to the fuzzy rule, having the largest weight. Another important aspect to consider in the proposed classifier is to define when a new rule should be created. This aspect is also an important contribution of our classifier. Contrary to the approach adopted by *eClass* [10] and *AutoClass* [13], TEDAClass does not use any global quantity (i.e., density) or all the learning data set. Instead, a new data cloud is created when the local typicality in a given instant of time for any rule, does not exceed some threshold which depends on the instant of time (corresponding to n sigma rule, for some given n). This condition is described as:

$$\forall R_i \in F t_i^k(\vec{x}) < T(k) \Leftrightarrow \eta(k) = 1. \tag{50}$$

Here $\eta(k) \in \{0, 1\}$ is a flag, raised, if the cluster should be created. Local typicality is defined in the same way but the summation is over the data points of a particular data cloud only [2].

After creating a new rule by the classifier, it is necessary to delete any of the already existing rules that are closer than the same threshold. This condition is defined as:

$$R_d = \{R_i \in F : t_i^k(\vec{x}_k) > T(k)\}. \tag{51}$$

As a summary, and taking into account the aspects previously explained, the formal description of the algorithms (training stage and recognition stage) is given below. It is important to emphasize again that the algorithm formulation is incremental and online, and it works sequentially from scratch, sample by sample.

Formal algorithm description - training stage:

1. Initialisation: $F = \oslash, k = 1$
2. For each new sample $\vec{x}_k$ do:
3. Calculate data sample local (per data cloud) normalised typicality $t_i^k(\vec{x}_k), i = [1 \dots N]$, N is the rule count for all the data samples.
4. $\forall R_i \in F t_i^k(\vec{x}) < T(k)$ create cloud with a centre in $\vec{x}_k$, initialise the parameters of the local typicality incremental learning (mean $\mu_{x,N+1}^k = \vec{x}_k$ and covariance $\Sigma_{x,N+1}^k$ as identity matrix, $N = N + 1$).
5. Calculate firing strength for every fuzzy rule according to the formula 49.
6. Increment the number of samples associated with the data, with respective fuzzy rule R_i with the highest typicality $t_i^k(\vec{x})$, calculated using the formula 36.

7. Update the *local* typicality parameters for the winning data cloud, $R_i(\vec{\mu}_{x,i}^k, \Sigma_{x,i}^k)$
8. Update the parameters matrix θ_i for the rule with the highest typicality $t_i^k(\vec{x})$ by the (fuzzily weighted) RLS algorithm [10]
9. Wait for a new sample and return to stage 2.

Formal algorithm description - recognition stage:

1. Initialisation: $F = \oslash, k = 1$
2. For each new sample $\vec{\hat{x}}_k$ do:
3. Calculate data sample local normalised typicality $t_i^k(\vec{\hat{x}}_k), i = [1 \dots N]$, N is the rule count for all the data samples, using the formula 36.
4. Calculate membership functions for each of the fuzzy rules according to the formula (44).
5. Calculate the regression $\hat{y}_i = \vec{\hat{x}}^k \Theta_i$ over $\vec{\hat{x}}^k$ for each fuzzy rule and calculate the weighted sum $\hat{y} = w_i^k(x)\hat{y}_i$
6. Take the winning class value for the weighted sum
7. Wait for sample $\vec{\hat{x}}^{k+1}$ to arrive

Finally, note that training and recognition stages can follow each other in an on-line implementation. In addition, since TEDAClass is recursive and one pass, it can be applied in many different real-time applications.

6. Experiments

To give an evidence of the proposed algorithm performance, we have conducted several experiments with very different data sets. It should be noted that although this algorithm is designed to be used in real-time, we have used several dataset to have comparable results with other classifiers. But this algorithm is totally developed to be applied in real-time.

6.1. DataSets

A. DataSets The first dataset, which is related with the symbol image recognition, is the well-known MNIST[24] dataset (Figure 1). This is a handwritten numeral database which contains 60,000 training and 10,000 test images of ten handwritten digits (0 to 9) with 28x28 pixels.

In addition, the method was checked on several datasets from various domains:

1. MAGIC data set [25] [26], [27], [28], a simulation data set for registration of high energy -particles;
2. ISOLET data set [29], [30], [31], [32], containing the vectors of features for isolated letters pronunciation for various speakers.
3. Poker hand data set [33], [34] for classification of the poker hands content. The data set is particularly interesting, because it contains integer data rather than real.

Figure 1: Experimental data samples (MNIST)

6.2. Results

To evaluate the performance of TEDAClass, it is compared with the following well-known classifiers:

- Neocognitron neural network[35];
- Fuzzy evolving classifier *eClass1*[10];
- Fuzzy evolving classifier *AutoClass1*[13].

We can see that the proposed classifier is also compared with the previously proposed evolving methods *eClass1* and *AutoClass1* to make sure that it performs well against evolving methods.

Tables 1 shows the classification rate used to evaluate the accuracy of the four classifiers for the MNIST[24] dataset. As one can see, different sizes of training sets (from 500 to 60000 samples) have been used.

Table 1: Recognition results comparison for MNIST[24] database

	Classifiers and classification rate (%)			
Training Set Size	*eClass1*	*AutoClass1*	*Neocognitron*	**TEDAClass**
500	93.26%	94.53%	92.36%	95,18%
1000	86.54%	95.82%	94.42%	95,92%
2000	96.42%	96.44%	96.04%	96,70%
3000	96.55%	96.50%	96.34%	96,67%
4000	96.62%	96.68%	96.62%	96,88%
5000	96.85%	96.91%	96.94%	97,16%
10000	97.19%	97.24%	-	97,38%
20000	97.32%	97.38%	-	97,53%
30000	97.46%	97.44%	-	97,68%
40000	97.45%	97.42%	-	97,66%
50000	97.46%	97.38%	-	97,65%
60000	97.46%	97.42%	-	97,63%

According to this data, we can see that proposed algorithm TEDAClass demonstrates significantly stronger results, achieving reasonable performance even for small amount of training data. *TEDAClass* algorithm, as well as *AutoClass1*, give more stable result, comparing with *eClass1*.The falls in performance of *eClass1* (i.e., for training set of 30000 samples gives better results than one of 40000 samples) can be explained by re-creation of some of the fuzzy rules, so that they did not have enough time to learn the patterns.

By the other hand, Table 2 shows the performance of *TEDAClass* on three benchmarking datasets (explained in the previous subsection). In this case, we can observe that if we take into account the MAGIC and Poker hand data sets, such modest results are caused by not so good separability of the data set. As it is described in [36], the goal of the dataset was to overcome threshold of accuracy 0.50122. The algorithm described there were tested on Waikato environment for knowledge analysis, and the best predictive accuracy, 0.56616, was given by *JRip* algorithm [37]. However, using *TEDAClass*, we obtain an accuracy of 0.5176.

Table 2: Performance of TEDAClass on some benchmarking datasets

	Data sets		
	MAGIC	ISOLET	Poker hand
Number of the classes	2	26	10
Data set size	19020	7797	1025010
Recognition rate for the data set	0.7773	0.9724	0.5016
Training data set size	9510	2599	25010
Testing data set size	9510	5198	1000000
Classification rate	0.7812	0.9007	0.5176

7. Conclusions and future work

In this paper a novel and promising approach is described for classification. This approach is based on the novel TEDA framework, giving a new systematic method to the online data analysis, and AnYa-type fuzzy rules. The description of this approach has been presented and the necessary novel derivations of the incremental formulae within TEDA framework have been also explained in detail.

The approach is evolving and fully recursive, and it has shown strong results against several well-known classification algorithms (neural networks, other fuzzy rule based systems). Unlike several previously proposed algorithms (*eClass*, *AutoClass*) it does not have any global metrics for the clusters set (i.e. global density), but relies only on the local metrics of the cluster. The experiments gave an evidence of practical applicability of the algorithm for classification in different domains: symbol recognition, isolated letters pronunciation, poker combinations, γ-particles registration event.

For all these domains, we have obtained superior or similar results to those provided by the state-of-the-art algorithms. Therefore, the algorithm can claim applicability. Moreover, as it is online and evolving, it has additional advantage to be capable of processing the data streams, and learning during the exploitation stage.

Future work will be oriented to deploying this classifier in a real environment in which the data streams is received obtained in real time. However, due to the flexibility of the proposed classifier, it can be used in different domains including banking and data mining for signal information (video, images, and sound). The new distances will be added for the textual and other domain-specific information types.

References

[1] P. Angelov, D. P. Filev, N. Kasabov, Evolving Intelligent Systems:Methodology and Applications, Wiley-IEEE Press, 2010.

[2] A. Plamen, Outside the box: an alternative data analytics framework, Journal of Automation, Mobile Robotics and Intelligent Systems 8 (2) (2014) 29–35.

[3] P. Angelov, R. Yager, A new type of simplified fuzzy rule-based system, International Journal of General Systems 41 (2) (2012) 163–185.

[4] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[5] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Review 65 (6) (1958) 386–408.

[6] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Inc., New York, NY, USA, 1995.

[7] D. S. Broomhead, D. Lowe, Radial basis functions, multi-variable functional interpolation and adaptive networks, Royal Signals and Radar Establishment Malvern.

[8] J. Pearl, Causality: Models, Reasoning, and Inference, Cambridge University Press, New York, NY, USA, 2000.

[9] P. Angelov, X. Zhou, Evolving fuzzy rule-based classifiers from data streams, IEEE Transactions on Fuzzy Systems: Special issue on Evolving Fuzzy Systems 16 (6) (2008) 1462–1475.

[10] L. Rokach, O. Maimon, Data Mining with Decision Trees - Theory and Applications, Vol. 69 of Series in Machine Perception and Artificial Intelligence, World-Scientific, 2007.

[11] P. Angelov, X. Zhou, F. Klawonn, Evolving fuzzy rule-based classifiers, in: Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on, 2007, pp. 220–225.

[12] P. P. Angelov, Autonomous learning systems: from data streams to knowledge in real-time, John Wiley And Sons Ltd, 2012.

[13] P. P. Angelov, D. Kangin, X. Zhou, D. Kolev, Symbol recognition with a new autonomously evolving classifier autoclass, in: 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems, EAIS, 2014, pp. 1–7.

[14] R. D. Baruah, P. P. Angelov, D. Baruah, Dynamically evolving fuzzy classifier for real-time classification of data streams, in: IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2014, 2014, pp. 383–389.

[15] E. Lughofer, E. Klement, FLEXFIS: A variant for incremental learning of takagi-sugeno fuzzy systems, in: IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2005, 2005, pp. 915–920.

[16] B. S. J. Costa, P. P. Angelov, L. A. Guedes, Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier, Neurocomputing 150 (2015) 289–303.

[17] N. Kasabov, V. Feigin, Z. Hou, Y. Chen, L. Liang, R. Krishnamurthi, M. Othman, P. Parmar, Evolving spiking neural networks for personalised modelling, classification and prediction of spatio-temporal patterns with a case study on stroke, Neurocomputing 134 (2014) 269–279.

[18] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man, and Cybernetics 15 (1) (1985) 116–132.

[19] E. H. Mamdani, S. Assilian, An experiment in linguistic synthesis with a fuzzy logic controller, International Journal of Man-Machine Studies 7 (1) (1975) 1–13.

[20] P. C. Mahalanobis, On the generalized distance in statistics, Proceedings of the National Institute of Sciences (Calcutta) 2 (1936) 49–55.

[21] P. Angelov, Anomaly detection based on eccentricity analysis, in: IEEE Symposium on Evolving and Autonomous Learning Systems, EALS, 2014, pp. 1–8.

[22] M. A. Woodbury, Inverting Modified Matrices, no. 42 in Statistical Research Group Memorandum Reports, Princeton University, Princeton, NJ, 1950.

[23] P. Angelov, D. Filev, Simpl_ets: a simplified method for learning evolving takagi-sugeno fuzzy models, in: IEEE International Conference on Fuzzy Systems, 2005, pp. 1068–1073.

[24] Y. LeCun, C. Cortes, MNIST handwritten digit database (2010).
URL `http://yann.lecun.com/exdb/mnist/`

[25] R. K. Bock, P. Savicky, MAGIC gamma telescope data set.
URL `archive.ics.uci.edu/ml/datasets/MAGIC+Gamma+Telescope`

[26] Methods for multidimensional event classification: a case study using images from a cherenkov gamma-ray telescope, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 516 (2) (2004) 511 – 528.

[27] P. Savicky, E. Kotrc, Experimental study of leaf confidences for random forest, Computational Statistics (2004) 1767–1774.

[28] J. Dvorak, P. Savickuy, Softening splits in decision trees using simulated annealing, in: Adaptive and Natural Computing Algorithms, Vol. 4431 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 721–729.

[29] M. Fanty, R. Cole, Spoken letter recognition, in: R. Lippmann, J. Moody, D. Touretzky (Eds.), Advances in Neural Information Processing Systems 3, Morgan-Kaufmann, 1991, pp. 220–226.

[30] T. G. Dietterich, G. Bakiri, Error-correcting output codes: A general method for improving multiclass inductive learning programs, in: in proceedings of AAAI-91, AAAI Press, 1991, pp. 572–577.

[31] T. G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, Journal of Artificial Intelligence Research 2 (1995) 263–286.

[32] R. Cole, M. Fanty, T. Dietterich, ISOLET data set (1994).
URL `https://archive.ics.uci.edu/ml/datasets/ISOLET`

[33] R. Cattral, F. Oppacher, D. Deugo, Evolutionary data mining with automatic rule generalization, Recent Advances in Computers, Computing and Communications (2002) 296–300.

[34] R. Cattral, F. Oppacher, Poker Hand data set.
URL `https://archive.ics.uci.edu/ml/datasets/Poker+Hand`

[35] K. Fukushima, Neocognitron for handwritten digit recognition, Neurocomputing 51 (2003) 161–180.

[36] R. Cattral, F. Oppacher, Discovering rules in the poker hand dataset, in: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO '07, ACM, New York, NY, USA, 2007, pp. 1870–1870.

[37] I. H. Witten, E. Frank, M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.