Czech Technical University in Prague

Faculty of Nuclear Sciences and Physical Engineering

DOCTORAL THESIS

Imperfect Learning of Multi-Classifier

Prague 2023 Radek Hřebík

Bibliografický záznam

Autor	Mgr. Ing. Radek Hřebík České vysoké učení technické v Praze Fakulta jaderná a fyzikálně inženýrská Katedra softwarového inženýrství
Název práce	Neperfektní učení multi-klasifikátoru
Studijní program	Aplikace přírodních věd
Studijní obor	Matematické inženýrství
Školitel	prof. Ing. Josef Jablonský, CSc. Vysoká škola ekonomická v Praze Fakulta informatiky a statistiky Katedra ekonometrie
Školitel specialista	doc. Ing. Jaromír Kukal, Ph.D. České vysoké učení technické v Praze Fakulta jaderná a fyzikálně inženýrská Katedra softwarového inženýrství
Akademický rok	2022/2023
Počet stran	161
Klíčová slova	klasifikátor, neperfektnost, skrytá třída, kritická citlivost, difuze

Bibliographic Entry

Author	Mgr. Ing. Radek Hřebík Czech Technical University in Prague Faculty of Nuclear Sciences and Physical Engineering Department of Software Engineering
Title of Dissertation	Imperfect Learning of Multi-Classifier
Degree Programme	Application of Natural Sciences
Field of Study	Mathematical Engineering
Supervisor	prof. Ing. Josef Jablonský, CSc. Prague University of Economics and Business Faculty of Informatics and Business Department of Econometrics
Supervisor Specialist	doc. Ing. Jaromír Kukal, Ph.D. Czech Technical University in Prague Faculty of Nuclear Sciences and Physical Engineering Department of Software Engineering
Academic Year	2022/2023
Number of Pages	161
Keywords	Classifier, Imperfectness, Hidden Class, Critical Sensitivity, Diffusion

Abstrakt

Disertační práce je zaměřena na konstrukci pokročilého klasifikátoru založeného na nové teorii. Tato nová teorie využívá velkého množství neperfektních metod učení, ať už ve formě učení s učitelem, nebo bez učitele, ke konstrukci tzv. skrytých tříd. Počet skrytých tříd se předpokládá vyšší než počet výstupních tříd klasifikační úlohy. Hlavní teorie spočívá v otázce optimálního sjednocování skrytých tříd tak, aby byla dosažena co nejvyšší přesnost, případně citlivost nového klasifikátoru. V práci se pro hodnocení kvality klasifikace využívá tzv. kritická citlivost, tedy nejnižší přesnost dosažená napříč třídami. Vzhledem k tomu, že nový klasifikátor využívá informace z těchto jiných systémů, tak tyto skryté klasifikátory musí sami za sebe určit příslušnost výsledné třídy původní úlohy. Tato příslušnost může být výsledkem neperfektního pokusu o klasifikaci do původních tříd, nebo výsledkem učení bez učitele, například shlukování. Díky tomuto přístupu je možné data reprezentovat pomocí názoru velkého množství různých klasifikátorů, které nemusí využívat všech vlastností ani vzorů z původních dat, může se jednat rovněž o jakékoli výsledky neperfektního učení. Součástí práce je návrh nového přístupu k samoorganizaci dat za využití difuze. Veškerá neperfektnost při vytváření skrytých tříd vede k rozmanitosti nového klasifikátoru. Nový postup se osvědčil na srovnání použití třinácti klasických metod klasifikace, kde bylo dosaženo zlepšení výsledků klasifikace hodnocené podle kritické citlivosti.

Abstract

The doctoral thesis focuses on the construction of an advanced classifier based on a new theory. This new theory is based on the use of a large number of imperfect learning methods, whether in the form of supervised or unsupervised learning, which are used to construct so-called hidden classes. The number of hidden classes is assumed to be higher than the number of output classes of the classification task. The main theory lies in the question of the optimal unioning of hidden classes in order to achieve the highest accuracy or sensitivity of the new classifier. The so-called critical sensitivity, i.e. the lowest accuracy achieved across classes, is used to evaluate the quality of classification in the thesis. Since the new classifier uses information from the other systems, these hidden classifiers themselves must determine the membership of the resulting class of the original task. This is obtained either as an imperfect attempt to classify into the original classes or as a result of unsupervised learning, such as clustering. This approach allows data to be represented by the results of a large number of different classifiers, which may not use all the properties or patterns of the original data, and may also include the results of the imperfect learning. Part of the thesis is focused on the design of a new approach to self-organization of data using diffusion. All imperfections in the formation of hidden classes lead to the diversity of the new classifier. The new procedure has been proven by comparing the use of thirteen traditional classification methods, where an improvement in the classification results evaluated according to critical sensitivity was achieved.

Acknowledgements

I would like to express my gratitude and thanks to Josef Jablonský for general support and many fruitful consultations regarding econometric aim specifications over the years. I would also like to thank Jaromír Kukal for providing a great deal of support, for his willingness to consult, and for a large number of stimulating ideas and valuable discussions over interesting topics over the years.

Radek Hřebík

Declaration

I confirm having prepared the thesis by my own and having listed all used sources of information in the bibliography.

Prague, 21 March 2023 Radek Hřebík

Contents

List of Symbols

Sets

$\mathbb{N}$ set of natural numbers

$\mathbb{R}$ set of real numbers

$\mathscr{S}$ pattern set

$\mathscr{C}$ output class

$\mathscr{H}$ hidden class

Variables

n number of features

m number of patterns

H number of hidden classes

N number of classes

D reduced dimension

$\mathbf{p}$ pattern vector

$\mathbf{X}$ input matrix

$\mathbf{y}$ classifier output vector

$\mathbf{y}^*$ desired output vector

y classifier output/response

y^* desired output value

$\mathbf{w}$ weight vector

w_0 Bias

Functions

c classifier

$\varkappa$ kernel function

Others

$\mathcal{H}^*$ Hilbert space

U uniform distribution

N multidimensional normal distribution

$\mathbb{I}$ unit matrix

List of Abbreviations

CLINK	Complete Linkage
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DWH	Data Whitening
EVD	Eigen Value Decomposition
iid	independent and identically distributed
k-NN	k-Nearest Neighbour algorithm
KRR	Kernel Ridge Regression
LDA	Linear Discriminant Analysis
MLP	Multi-Layer Perceptron
MM	Max-Margin method
OUHC	Optimal Union of Hidden Classes
PCA	Principal Component Analysis
QDA	Quadratic Discriminant Analysis
QE	Quantization Error
RAO	Multiclass Discriminant Analysis by RAO
RBF	Radial Basis Function
red	reduced
RR	Ridge Regression
RVFL	Random Vector Functional Link
SLINK	Single Linkage
SOM	Self-Organizing Map
TE	Topographic Error
xtd	extended

Introduction

The thesis is focused on a novel imperfect approach to classification tasks. The importance of such a task remains indisputable in the current world. The classification remains important over time and in the current environment of ubiquitous artificial intelligence. The pressure on quick and somehow right decisions is still here and it is not likely to disappear in the near future. The decision-making process is one of the main components of artificial intelligence across scientific disciplines.

In the first chapter, we provide a summary of classification methods in the vector space. We presented mainly the methods of the perfect and the imperfect learning where we expect a potential improvement by using novel learning strategies. The main idea is that the imperfectness could lead to better results, therefore improve the traditional, perfect, learning strategies. We do not search for any extraordinarily good methods, but we focus on relatively good and well known algorithms that could be improved.

The second chapter is a summary of the main goals of the thesis leading to a novel imperfect classifier.

In the third chapter, we focus on novel methods for self-organization. We derive our new method from the traditional Kohonen learning. The aim is to benefit from the biological processes which can be captured using mathematical equations.

The main idea of the thesis is presented in the fourth chapter, which deals with the core contribution of the thesis: a novel theory of the imperfect learning. We aim to formulate the imperfect learning strategy by using the linear programming theory.

The fifth chapter is focused on the structure of the proposed classifier. We summarize the main options for the data preprocessing and the forming of hidden classes. The hidden classifier based on the optimal union of hidden classes is a part of the novel classifier scheme. The approaches for the classifier validation are also discussed in this chapter.

The experimental part is focused on a detailed analysis of the proposed classifier. Firstly, we discuss one traditional dataset in detail. Secondly, we compare the results of the new proposed classifier with the traditional perfect learning strategy by classification of nine additional datasets. We present also the results on two specific real datasets.

The implementation of the new classifier is in the Matlab. Therefore, the basic structure of the proposed classifier and its organization to basic functions is briefly described in the last chapter.

In the first appendix, we present a brief summary of the basic Matlab functions. The second appendix presents three main publications related to our research in the impact journals.

Chapter 1

Classification Methods in Vector Space

We start this chapter with a definition of the classification task. The presented notation will be provided throughout the whole thesis. The learning techniques can be divided into supervised and unsupervised. As a starting point of data analysis, preprocessing techniques will be presented in this section. Last but not least, validation approaches will be discussed.

1.1 Classification Task

Basic facts related to the classification into several classes are summarized in this section. The basic frame of the vector patterns classification [1] into several classes is established first. Let $n, m, N \in \mathbb{N}$ be the number of features, patterns, and classes satisfying $N \geq 2$. Let $\mathbf{x} \in \mathbb{R}^n$ be the feature vector and $y, y^* \in \{1, ..., N\}$ be the classifier output and its required value, denoting a pattern as $\mathbf{p} = (\mathbf{x}, y^*)$, the classifier is defined as a function

$$\mathrm{c} : \mathbb{R}^n \to \{1, ..., N\} \tag{1.1}$$

and the classifier response is, therefore, $y = \mathrm{c}(\mathbf{x})$. Denoting $\mathbf{x}_k \in \mathbb{R}^n$, $y_k^* \in \{1, ..., N\}$ as the feature vector and the given output of the k-th pattern, we define the pattern set as

$$\mathscr{S} = \{(\mathbf{x}_k, y_k^*) : k = 1, ..., m\}. \tag{1.2}$$

The pattern set can be represented by any input matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and any output vector $\mathbf{y}^* \in \{1, ..., N\}^m$. Any classifier is a complex system that applies various data processing techniques to obtain the final decision. Selected approaches are summarised in the following sections.

The basic machine learning techniques used for the classification differ in the need for previous knowledge of data classification [1]. Therefore we can distinguish between two main learning techniques: the supervised and the unsupervised techniques. The supervised learning means that we work with labelled data. When we do not have any previous knowledge of the class identification, we talk about the unsupervised learning over unlabelled data. The artificial neural network models are very popular tools for the classification. They can be based on both the supervised and the unsupervised learning techniques.

The supervised techniques are further divided into density-based, linearity-based, and iterative ones. Pattern clustering and self-organizing maps represent unsupervised techniques.

1.2 Supervised Learning Techniques

The supervised techniques [2] require a labelled dataset. The task [2] of the supervised learning is based on pairs of input-output data. If the algorithm tries to label the input into two distinct classes, it is called the binary classification. Selecting between more than two classes is referred to as the multi-class classification. We will start with the traditional techniques based on the density and linear combination. The third supervised learning technique will be focused on the iterative learning.

1.2.1 Density Based Classification

There are many methods that are based on the model of multidimensional Gaussian normal distribution. The main idea is that every output class is supposed to have a feature vector belonging to the normal distribution as

$$\mathbf{X} \sim \mathrm{N}(\mu, \Sigma) \tag{1.3}$$

where $\mathbf{x}$ is the feature vector, μ is the center of the class $\mathbf{x}, \mu \in \mathbb{R}^n$ and $\Sigma \in \mathbb{R}^{n \times n}$ and $\Sigma > 0$ is the positive definite covariance matrix. The adequate formula for the individual class density is

$$f(\mathbf{x}, \mu, \Sigma) = (2\pi)^{-n/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)\right). \tag{1.4}$$

But we have N classes denoted as $\mathscr{C}_k$ for $k = 1, ..., N$ and the number of patterns in these classes is denoted as m_k, $m_k \geq 2$ and $\sum_{k=1}^{N} m_k = m$. We calculate the gravity center of every class as

$$\mathbf{e}_k = \frac{1}{m_k} \sum_{j=1}^{m_k} \mathbf{x}_{k,j} \tag{1.5}$$

and also the estimate of the covariance matrix as

$$\mathbf{C}_k = \frac{1}{m_k - 1} \sum_{j=1}^{m_k} (\mathbf{x}_{k,j} - \mathbf{e}_k)(\mathbf{x}_{k,j} - \mathbf{e}_k)^T. \tag{1.6}$$

In some applications, we also design the central covariance matrix which is defined as the weighted sum of the individual covariance matrices

$$\mathbf{C}^* = \frac{1}{m} \sum_{k=1}^{N} m_k \mathbf{C}_k. \tag{1.7}$$

Linear Discriminant Analysis (LDA)

The hypothesis of the multivariate normal distribution is used in the regularized form of linear discriminant analysis [3] as follows. We have a regularization parameter $\lambda \geq 0$. In the given point $\mathbf{x}$ and class k we calculate the density using the central covariance matrix and the density (1.4)

$$f_k = f(\mathbf{x}, \mathbf{e}_k, \mathbf{C}^* + \lambda \mathbb{I}) \tag{1.8}$$

for $k = 1, ..., N$ where $\mathbb{I} \in \mathbb{R}^{n \times n}$ is identity matrix.

After the class density evaluation, we decide whether the given pattern $\mathbf{x}$ belongs to the j-th class, $\mathbf{x} \in \mathscr{C}_j$, or not. When $f_j > f_k$ for all $k \neq j$ we say that $\mathbf{x} \in \mathscr{C}_j$, therefore $y = j$. In any case, we say that $y = 0$ as a symbol of unknown classifier output.

The role of the regularization parameter λ is cardinal. When $\lambda = 0$ the regularization is omitted and the traditional LDA is realized. We use positive $\lambda > 0$ whenever the covariance matrix $\mathbf{C}^*$ is degenerated. The optimal value of λ will be determined experimentally.

Quadratic Discriminant Analysis (QDA)

A similar technique is used in the advanced method of quadratic discriminant analysis [4]. The method differs only in the covariance matrix where every class has its own center of gravity and also its covariance matrix

$$f_k = f(\mathbf{x}, \mathbf{e}_k, \mathbf{C}_k + \lambda \mathbb{I}) \tag{1.9}$$

for , $k = 1, ..., N$ and the density (1.4).

Parzen Estimate

Another possibility of calculating the class density f_k is to use the Gaussian normal distribution again but with raw data instead of the center of gravity and by

averaging the individual densities [5]. Resulting formula is

$$f_k = \frac{1}{m_k} \sum_{j=1}^{m_k} f(\mathbf{x}, \mathbf{x}_{k,j}, \mathbf{H}_k), \tag{1.10}$$

where $\mathbf{x}_{k,j}$ is the symbol of the j-th vector of the k-th class and the positive semidefinite matrix $\mathbf{H}_k \in \mathbb{R}^{n \times n}, \mathbf{H} > 0$ is subject to the experimental research again. There are two traditional rules of thumb for setting this matrix. The matrix is designed as positive definite but diagonal only.

Therefore, $h_{i,j} = 0, i \neq j$ and in the case of Scott's rule [6] we calculate

$$h_{k,i,i} = \sigma_{k,i}^2 m_k^{-2/(n+4)}. \tag{1.11}$$

The alternative choice is Silverman's rule [7] where

$$h_{k,i,i} = \sigma_{k,i}^2 m_k^{-2/(n+4)} \frac{4}{n+2}^{2/(n+4)}. \tag{1.12}$$

where $\sigma_{k,i}^2$ is the estimate of variance of the i-th coordinate within the k-th class.

Loftsgaarden-Quesenberry Estimate (LQ)

In case we do not prefer the Gaussian normal distribution we can use the point-to-point Euclidean distances to design the classifier in an alternative way [8, 9]. First, we calculate the distance of the input vector to all vectors of the class k as

$$d_{k,j} = \|\mathbf{x} - \mathbf{x}_{k,j}\| \tag{1.13}$$

for $j = 1, ..., m_k$. Based on the author's suggestion, we also set the critical parameter

$$r_k = \lceil m_k^{1/2} \rceil \tag{1.14}$$

for every class. After denoting the V_n as the volume of n-dimensional unit ball we directly calculate the class density as

$$f_k = \frac{r_k - 1}{m_k V_n d_{k,r_k}^n}, \tag{1.15}$$

where d_{k,r_k}^n is the r_k-th smallest distance in the k-th class.

k-Nearest Neighbour (k-NN) Estimate

The last presented density-based classifier is the k-NN classifier which is also based on the Euclidean distances. The main idea of this alternative approach to the density estimation is to fix the number of points required to estimate the

density. This allows the volume of the enclosing region to accommodate this fixed number of points [10].

We also calculate the distances between the input vector $\mathbf{x}$ and all vectors of overall classes. According to

$$\rho_i = \|\mathbf{x} - \mathbf{x}_i\| \tag{1.16}$$

for $i = 1,...,m$ and after sorting the distances, we evaluate the k-th distance as

$$\rho^* = \rho_{(k)}. \tag{1.17}$$

Instead of the densities, we calculate only the cardinality in the ρ^* neighbourhood for every class as

$$f_j = \text{card}\{i: \rho_i \leq \rho^* \,\&\, y_i^* = j\} \tag{1.18}$$

for $j = 1,...,N$.

The final decision about the class membership uses the same algorithm as in the case of the LDA.

1.2.2 Linearity Based Classification

Presented techniques are based on the linear combination of pattern properties. Some of them are applicable only to the linearly separable pattern sets.

The linear techniques are based on the weight vector $\mathbf{w} = (w_0, w_1, ..., w_n)$ whose dimension is $n + 1$ and which is used in the linear optimization tasks. The w_0 represents a bias.

Max-Margin (MM) Classifier

The abilities of the linear classification can be demonstrated as a direct application of Max-Margin classifier [11]. Traditionally, this classifier is applied only for $N = 2$ using the bipolar notation i.e. $\mathbf{X} \in \mathbb{R}^{m \times n}, \mathbf{y}^* \in \{\pm 1\}^m$ where $\mathbf{y}^* = +1$ represents $\mathscr{C}_1$ and $\mathbf{y}^* = -1$ represents $\mathscr{C}_2$. The bipolar response is driven by the formula

$$y = \text{sign}\left(w_0 + \sum_{j=1}^{n} w_j x_j\right) \tag{1.19}$$

where $\mathbf{w} \in \mathbb{R}^{n+1}$ is an unknown vector satisfying

$$\left(w_0 + \sum_{j=1}^{n} w_j x_{i,j}\right) y_i^* \geq 1 \tag{1.20}$$

for all $i = 1,...,m$ as separability conditions. Using the maximum margin condition

$$\sum_{j=1}^{n} w_j^2 = \min, \tag{1.21}$$

the optimum weights are the solution of the quadratic optimization task.

The Max-Margin classifier is applicable only to a linear separable pattern set.

The Max-Margin technique can also be used for the optional number of output classes N in this way. We perform N Max-Margin classifications using the previous procedure for every class. The union of rest classes is used as the opposite one. Finally, we calculate

$$f_k = w_{k,0} + \sum_{j=1}^{n} w_{k,j} x_j \tag{1.22}$$

for every class. The final decision is obtained by the maximization of f_k.

Ridge Regression (RR)

The basic approach in regression analysis is represented by the least squares method. The well-known method approximates the solution of overdetermined systems by minimizing the sum of the squares of the residuals made in the results of every single equation.

To face the negative impact of the collinearity on the least squares estimator we can use the Ridge Regression [12], which provides a means of addressing the problem of the collinearity without removing variables from the original set of independent variables.

The relation to generalized inverse regression was presented and the RR was confirmed to be a safe procedure for selecting variables and producing coefficients that predict and extrapolate better than least squares, as summarized in [13].

It is necessary to extend the matrix $\mathbf{X}$ first using the first column for the bias input as

$$\mathbf{X}_{\text{xtd}} = (\mathbf{I}|\mathbf{X}) \in \mathbb{R}^{m\times(n+1)}. \tag{1.23}$$

When we plan the classification to N classes we also use matrix $\mathbf{Y}^*$,

$$\mathbf{Y}^* \in \{0,1\}^{m\times(n+1)} \tag{1.24}$$

satisfying

$$\sum_{j=1}^{N} y^*_{i,j} = 1 \text{ for } i = 1,...,m, \tag{1.25}$$

which represents the original vector $\mathbf{y}^*$ in an equivalent form.

The result of learning of the RR is the matrix

$$\mathbf{W} = \left(\mathbf{X}^{\text{T}}_{\text{xtd}}\mathbf{X}_{\text{xtd}} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\text{T}}_{\text{xtd}}\mathbf{Y}^*, \tag{1.26}$$

where λ is a positive regularization parameter that is subject to efficient cross-validation.

If we apply the weights potentially to another pattern set we use the formula for real output first as

$$\mathbf{Y} = \mathbf{X}_{\text{xtd}}\mathbf{W} \in \mathbb{R}^{m\times N}. \tag{1.27}$$

The classifier into N classes makes a decision on the best output using the formula

$$c_i \in \underset{j=1,\dots,N}{\arg\max}\, y_{i,j} \tag{1.28}$$

and therefore, the vector $\mathbf{c} \in \{1,\dots,N\}^m$. There are many approaches to the selection of the class using the vector values of the matrix $\mathbf{Y}$ but the mentioned approach will be preferred in the experimental part.

Kernel Ridge Regression (KRR)

The kernel ridge regression is a special case of the support vector regression, which has been known in the Bayesian statistics for a long time [14].

Every kernel method is based on the feature mapping

$$\Phi\colon \mathbb{R}^n \to \mathcal{H}^* \tag{1.29}$$

where $\mathbb{R}^n$ is the original space of patterns and $\mathcal{H}^*$ is a Hilbert space. There are many operations permitted in Hilbert space but the kernel methods are based only on the linearity and the existence of the scalar product. The scalar product in the Hilbert space is defined as

$$\varkappa(\mathbf{x},\mathbf{y}) = \Phi^{\mathrm{T}}(\mathbf{x})\Phi(\mathbf{y}) \tag{1.30}$$

and it is called the Kernel function [2].

When we have a pattern set of size m we calculate the Gram matrix as

$$\mathbf{K} = \left(\varkappa(\mathbf{x}_i,\mathbf{x}_j)\right)_{i,j=1}^{m}. \tag{1.31}$$

There are many possibilities to design the function $\varkappa$ but in the majority of application the authors use the Gaussian kernel $\varkappa(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2}\right)$.

Having the input vector $\mathbf{x}$ we also calculate an adequate feature vector in the Hilbert space as

$$\mathbf{k} = (\varkappa(\mathbf{x},\mathbf{x}_i))_{i=1}^{m} \tag{1.32}$$

The KRR [15, 14] yields from the theory of the RR and operates with the regularization parameter $\lambda > 0$. Resulting formula is

$$\mathbf{y} = (\mathbf{y}^*)^{\mathrm{T}}(\mathbf{K} + \lambda\mathbb{I})^{-1}\mathbf{k}. \tag{1.33}$$

Therefore, the traditional Kernel Ridge Regression has two parameters σ and λ which can rapidly change its properties.

Random Vector Functional Link (RVFL)

The Random Vector Functional Link network is a kind of the multilayer perceptron. The output weights are chosen as adaptable parameters and the remaining parameters are fixed and randomly generated as iid. The RVFL is an efficient universal approximator that avoids the curse of dimensionality [16].

The feature space of dimension n is extended by $J \in \mathbb{N}$ hidden neurons which are represented by the traditional hyperbolic tangent perceptrons with the weights of maximal value $v_{\max}$.

The adequate matrix of the supporting weights $\mathbf{V} \in [-v_{\max}, +v_{\max}]_J^{J\times(n+1)}$ is generated using the uniform distribution as $v_{i,j} \backsim \mathrm{U}([-v_{\max}, +v_{\max}])$.

The adequate responses of hidden neurons for every pattern are collected in the matrix $\mathbf{H} \in (-1,+1)^{m\times J}$, which is calculated as

$$\mathbf{H} = \tanh(\mathbf{X}_{\mathrm{xtd}}\mathbf{W}^{\mathrm{T}}) \tag{1.34}$$

where the $\mathbf{X}_{\mathrm{xtd}}$ is the original matrix $\mathbf{X}$ extended by unit raw for the bias. The hidden response matrix $\mathbf{H}$ is used for the construction of a modified matrix

$$\mathbf{X}_{\mathrm{RVFL}} = (\mathbf{X}_{\mathrm{xtd}}|\mathbf{H}) \tag{1.35}$$

satisfying

$$\mathbf{X}_{\mathrm{RVFL}} \in \mathbb{R}^{m\times(n+1+J)}. \tag{1.36}$$

Finally, we use the ordinary RR, as described above, to calculate the output weights as

$$\mathbf{W} = \left(\mathbf{X}_{\mathrm{RVFL}}^{\mathrm{T}}\mathbf{X}_{\mathrm{RVFL}} + \lambda\mathbb{I}\right)^{-1}\mathbf{X}_{\mathrm{RVFL}}^{\mathrm{T}}\mathbf{Y}^{*}. \tag{1.37}$$

1.2.3 Iterative Learning

The iterative learning of classifiers is a traditional way to obtain unknown weights of a given classifier using the stochastic and the deterministic techniques. First, we assume that the classifier has p parameters, where $p \in \mathbb{N}$. Therefore, we can formally denote the classifier weights vector as $\mathbf{w} \in \mathbb{R}^p$, regardless of whether the structure of the classifier is hierarchical or matrix. In the traditional iterative learning the algorithms are based on the penalty function as

$$\varphi : \mathbb{R}^n \times \{1,...,N\} \times \mathbb{R}^p \to \mathbb{R}_0^+. \tag{1.38}$$

Using n patterns, we can formulate the total penalty function

$$\Phi(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m}\varphi(\mathbf{x}_i, y_i^*, \mathbf{w}), \tag{1.39}$$

which is subject to minimization according to unknown weights vector $\mathbf{w}$. This total penalty can be formulated also in the statistical sense using the mean value operator E as

$$\Phi(\mathbf{w}) = \mathrm{E}\varphi(\mathbf{x}, y^*, \mathbf{w}). \tag{1.40}$$

The traditional gradient methods begin with an initial estimate of weight $\mathbf{w}_0 \in \mathbb{R}^p$ and use $\lambda > 0$ as the learning rate parameter. The short step gradient method for local minimum finding is based on the iterative formula

$$\mathbf{w}_k = \mathbf{w}_{k-1} - \lambda \frac{\partial \Phi(\mathbf{w}_{k-1})}{\partial \mathbf{w}}. \tag{1.41}$$

There are two disadvantages of gradient methods. First, when the convergence is achieved using a very small value of λ parameters and many steps of learning, there is no guarantee that the global optimum of function Φ will be found. The second disadvantage is the higher time complexity caused by the computing time proportionality to the number of patterns.

Therefore, we generally prefer the stochastic gradient methods, which are based on the idea that the mean value of φ penalty is approximated by the individual value of a randomly selected pattern. First, we introduce the set of pattern indices $\mathcal{I}^* = \{1, ..., m\}$, then we select randomly and uniformly the index in the k-th step

$$i_k \sim \mathrm{U}(\mathcal{I}^*). \tag{1.42}$$

Finally, we adopt the weights using the formula

$$\mathbf{w}_k = \mathbf{w}_{k-1} - \lambda \frac{\partial \varphi(\mathbf{x}_{i_k}, y^*_{i_k}), \mathbf{w}_{k-1})}{\partial \mathbf{w}}. \tag{1.43}$$

The main advantage of the stochastic approach is in its time complexity reduced by a factor m but these techniques also find the local minimum of Φ instead of the global minimum.

In general, the Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) neural network are good examples of the iterative learning networks.

The MLP neural network represents a class of feed-forward artificial neural networks [17] and it can also distinguish the linearly non-separable patterns. The MLP training can be performed by using the optimizer [18] reaching a high level of accuracy in the classification and the approximation.

The RBF networks [19] are used as an alternative to the two-layer MLP neural network. The RBF learning algorithm is based on a random setting of the RBF centers and it is followed by solving the weight problem using the singular-value decomposition.

The presence of an untrained class and the comparison [20] of the MLP and the RBF neural networks lead to the identification and the exclusion of some cases of the untrained classes resulting in an increase in the classification accuracy.

1.3 Unsupervised Learning Techniques

The extraction of features from unlabelled data represents an unsupervised technique. In this case, the model works on its own to discover the patterns. There is

no need for the class representation. These techniques are represented mainly by numerous clustering techniques.

1.3.1 Pattern Clustering

The typical clustering models [21] are connectivity-based, centroid-based, distribution, density, subspace, graph-based, and neural network ones. Well known unsupervised neural network model is represented by the Self-Organizing Maps (SOM). We can also include subspace models as the Principal Component Analysis (PCA) or the Independent Component Analysis which can be also used as the data preprocessing techniques for the dimension reduction [22] of the original dataset consisting of a large number of interrelated variables.

One of the unsupervised algorithms is called the K-means, or the easy clusterization. This method is based on the grouping of the patterns into groups based on their characteristics. The clusters are made by minimizing the sum of the distances between each pattern and the center of a cluster. We select the number of centroids that are established in the data space, for instance, choosing them randomly. Each pattern is assigned to its nearest centroid. In the next step we update the centroids. The position of the centroid of each group is updated, taking the average position of the objects belonging to said group as the new centroid [1].

There are also various approaches to the pattern classification. In this subsection, we focus on the sequential clustering algorithms, such as the SLINK, the CLINK, and finally the DBSCAN. The SLINK algorithm [23] carries out the single-link cluster analysis on an arbitrary dissimilarity coefficient and it provides a representation of the resultant dendrogram, which can be readily converted into the usual tree diagram.

SLINK

The SLINK algorithm uses the positive parameter ϵ to select the pattern interconnection using the condition $d_{i,j} = d(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon$, where d is a distance in $\mathbb{R}^n$, e. g. the Euclidean distance. The pattern pairs satisfying this condition are connected by an edge in the sense of the undirected graph theory. The final structure of the undirected graph obviously depends on the data and the parameter ϵ. And every component of the resulting graph, which consists of at least two vertices, is defined here as a cluster. The rest of the patterns are defined as outliers. The traditional algorithm of Sibson [24] has the time complexity $O(m^2)$, which is problematic for large pattern sets.

There is also an alternative implementation [25], which comes from a reduction in the number of distance calculations required by the standard implementation of the SLINK with the time complexity $O(m \log m)$ in the case of m patterns. Hierarchical clustering, which omits the initial sorting and the consecutive clustering and which has a linear time complexity as an alternative to the single linkage clustering, has also been presented [26].

CLINK

An algorithm for a complete linkage clustering the CLINK [27] is based, just as the SLINK, on the compact representation of a dendrogram. The CLINK algorithm is slightly complicated but it forms more compact clusters in general. First, we denote the complete graph of n vertices as $\mathcal{K}_n$, where the vertices are fully interconnected. Every component which is isomorphic with $\mathcal{K}_n$ and consist of at least two points, is the cluster of the CLINK. The remaining vertices are outliers. But the time complexity of the original algorithm is also $O(m^2)$.

The fast algorithms [28] for the CLINK clustering show that the complete linkage clustering of m points can be computed in $O(m \log^2 m)$ time.

DBSCAN

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [29, 10] represents a non-parametric algorithm with a given set of points in the metric space and it groups together the points that are closely packed together and mark outliers.

We will study patterns in the vector space as $\mathbf{x}_k \in \mathbb{R}^n, k = 1, ..., m$, where m, n represent the number of patterns and the space dimensionality, respectively, but the DBSCAN is defined in the metric space. After the application of the Euclidean distance we can define the mutual distances as $d_{i,j} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$. Various versions of this algorithm [30, 31] differ in the method of the distance computation. The inefficient implementations [32, 33] calculate all mutual distances before the data clustering with the time complexity $O(n^2)$ but there are more effective procedures that rapidly decrease the time complexity of the DBSCAN to $O(m \log m)$ as in the case of the SLINK.

The DBSCAN is driven by two parameters $\epsilon > 0$, $k_{\min} \geq 2$, which fully depend on the user's opinion. We will set them to obtain the best sensitivity of the resulting classifier in the process of cross-validation. The DBSCAN generates an undirected graph $\mathcal{G}$ with the vertex set $\mathcal{V} = \{1, ..., m\}$ and the edge set $\mathcal{E} = \{e_1, e_2, ..., e_t\}$ and the pattern $\mathbf{x}_i$ is placed in the vertex i for $i = 1, ..., m$. There are three types of vertices: a hard member, a soft member, and an outlier.

The vertex i is called the hard member when $\text{card}\{j : d_{i,j} \leq \epsilon\} \geq k_{\min}$. Every edge $e = \{i, j\}$ has to satisfy $d_{i,j} \leq \epsilon$. The edge e is called the hard connection when the vertices i, j are the hard members. A soft connection is the edge e, where the node i is the hard member but the node j is not. The remaining edges are eliminated. The resulting graph $\mathcal{G}$ has several components. The component is declared as a cluster when it has two vertices at least. The remaining discrete components are declared as outliers.

The main advantage of the SLINK, the CLINK, and the DBSCAN is the capacity for the sequential learning with acceptable time complexity. The hard members of the DBSCAN, number of clusters, and outliers are invariant to pattern order during the learning process.

1.3.2 Self-Organizing Map

The Self-Organizing Map (SOM) is a traditional tool for the data analysis that transforms the data patterns from the input space into the vertices of an undirected SOM graph with a given topology and unit-length edges. The input patterns are from the metric vector space in many applications. The parameters of the SOM are the weights that are placed into the vertices and they are the subject of learning. The Kohonen learning [34] is the first approach frequently used in many applications [35, 36, 37, 38].

Kohonen Learning

The Kohonen network maps the input vectors (patterns) of arbitrary dimension n into the vertices of a given undirected graph. One of the main expected results is that patterns close to one another in the input space are to be close to one another in the graph. This is called to be topologically ordered. A Kohonen network is composed of a grid of the output units and N input units. The input pattern is fed to each output unit. The input lines to each output unit are weighted. These weights are initialized to small random numbers.

In the case of the Kohonen learning [39] we use rules as follows. The weight of the i-th neuron is changed in the q-th step by the rule

$$\mathbf{w}_i(q) = \mathbf{w}_i(q-1) + \alpha(q) \cdot g_{i,q} \cdot (\mathbf{x}_q - \mathbf{w}_i(q-1)) \tag{1.44}$$

for $i = 1, ..., H$, $\mathbf{x}_q \sim \mathrm{U}(\mathcal{S})$ is the uniformly selected pattern from $\mathcal{S}$, $g_{i,q}$ is the spatial gain and $\alpha(q) > 0$ is the ageing function, which is supposed to be non-increasing. The winner selection process is called the Kohonen rule [39]

$$\varphi_q \in \underset{k=1,...,H}{\arg\min} \|\mathbf{x}_q - \mathbf{w}_k\|_2. \tag{1.45}$$

In the traditional SOM learning, we have to initialize the weights as small random iid [40] and use the appropriate ageing strategy.

1.4 Preprocessing Techniques

The first but optional step of any classification is an efficient transformation, that decreases the number of features but saves information about the pattern differences.

1.4.1 Data Whitening (DWH)

The main idea of the Principal Component Analysis (PCA) [41] is to reduce the dimensionality of the original dataset consisting of a large number of interrelated

variables. The reduction retains the variation present in the data set as much as possible. The aim is achieved by transforming into a new set of variables called principal components. These components are uncorrelated and ordered so that the first few retain the most of the variation present in all of the original variables.

Let $D \in \mathbb{N}$ be a reduced dimension satisfying $D < n$. The dimensionality reduction from $\mathbb{R}^n$ to $\mathbb{R}^D$ using the PCA is based on a linear transformation

$$\mathbf{z} = \mathrm{PCA}(\mathbf{x}) = \mathbf{W}_1^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0). \tag{1.46}$$

The PCA is designed to satisfy $\mathrm{E}\,\mathbf{z} = \mathbf{0}$ and $\mathrm{var}\,\mathbf{z} = \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix. Resulting parameters of the PCA are

$$\mathbf{W}_1 \in \mathbb{R}^{n \times D} \tag{1.47}$$

and

$$\mathbf{x}_0 = \frac{1}{m} \sum_{k=1}^{m} \mathbf{x}_k. \tag{1.48}$$

The transforming matrix $\mathbf{W}_1$ is calculated as follows. First, we shift the input matrix to obtain $\mathbf{X}_{\mathrm{S}} = \mathbf{X} - \mathbf{1}_m \mathbf{x}_0^{\mathrm{T}}$, where $\mathbf{1}_m$ is m-dimensional vector of units. Then we calculate the covariance matrix $\mathbf{A} = \mathbf{X}_{\mathrm{S}}^{\mathrm{T}} \mathbf{X}_{\mathrm{S}} \geq 0$ and apply the Eigen Value Decomposition (EVD) to find of eigenvalues $\mathbf{v} \in \mathbb{R}^n$ and eigenvectors $\lambda \geq 0$ in equation $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{v} = \mathbf{0}$ where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is an identity matrix.

The solutions can be ordered as $\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(D)} \geq 0$ with the corresponding normalized eigenvectors $\mathbf{v}_{(1)}, \mathbf{v}_{(2)}, ..., \mathbf{v}_{(D)}$. The resulting PCA matrix [41] is

$$\mathbf{W}_1 = (\mathbf{v}_{(1)}, \mathbf{v}_{(2)}, ..., \mathbf{v}_{(D)}) \tag{1.49}$$

and the dimensionality reduction generates a new feature matrix

$$\mathbf{Z} = \mathbf{X}_{\mathrm{S}} \mathbf{W}_1. \tag{1.50}$$

The data whitening (DWH) [42, 43] represents an improved process of the PCA, which guarantees the unit covariance matrix of the resulting vector. The transform is defined as

$$\mathbf{z} = \mathrm{DWH}(\mathbf{x}) = \mathbf{W}_2^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0). \tag{1.51}$$

The matrix $\mathbf{W}_2^{\mathrm{T}}$ is designed to satisfy $\mathrm{E}\,\mathbf{z} = \mathbf{0}$ and $\mathrm{var}\,\mathbf{z} = \mathbb{I}_n$. Using the result of the EVD we directly calculate [42]

$$\mathbf{W}_2 = \left(\frac{\mathbf{v}_{(1)}}{\sqrt{\lambda_{(1)}}}, \frac{\mathbf{v}_{(2)}}{\sqrt{\lambda_{(2)}}}, ..., \frac{\mathbf{v}_{(D)}}{\sqrt{\lambda_{(D)}}} \right). \tag{1.52}$$

Due to duality, we can perform the data whitening for $m < n$ in a more efficient way. We calculate $\mathbf{B} = \mathbf{X}_S\mathbf{X}_S^T \geq 0$ and perform its EVD. The resulting EVD equation is $(\mathbf{B} - \lambda\mathbf{I}_m)\mathbf{u} = \mathbf{0}$. The solutions can be ordered again as

$$\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(D)} > 0 \tag{1.53}$$

with the corresponding normalized eigenvectors $\mathbf{u}_{(1)}, \mathbf{u}_{(2)}, ..., \mathbf{u}_{(D)} \in \mathrm{R}^m$. The resulting whitening matrix is

$$\mathbf{W}_2 = \mathbf{X}_S^T\left(\frac{\mathbf{u}_{(1)}}{\lambda_{(1)}}, \frac{\mathbf{u}_{(2)}}{\lambda_{(2)}}, ..., \frac{\mathbf{u}_{(D)}}{\lambda_{(D)}}\right) \tag{1.54}$$

and the data whitening generates a new feature matrix $\mathbf{Z} = \mathbf{X}_S\mathbf{W}_2$ in both cases. The data whitening in its primal or dual form is preferred in this paper for the optional data preprocessing.

1.4.2 Multiple Discriminant Analysis

Another approach to the dimensionality reduction is based on the knowledge of class membership. Having information about the classes we can also perform the linear data transformation to obtain a higher data separation. The classical Fisher discriminant analysis [44, 45] is designed for two classes but the RAO method [1, 46] generalized it for a multi-classification task as follows.

The RAO method transforms the data from $\mathbb{R}^n$ to $\mathbb{R}^{N-1}$ for $N \geq 2$ using a linear transformation

$$\mathbf{z} = \mathrm{RAO}(\mathbf{x}) = \mathbf{W}_3^T(\mathbf{x} - \mathbf{x}_0), \tag{1.55}$$

where $\mathbf{W}_3 \in \mathbb{R}^{n\times(N-1)}$.

The method is based on the pattern index sets $\mathcal{D}_i \in \{k \in \mathbb{N} : y_k^* = i\}$ for $i = 1, ..., N$ and their cardinalities $m_i = \mathrm{card}\,\mathcal{D}_i$. After the evaluation of the cluster centres

$$\mathbf{t}_i = \frac{1}{m_i}\sum_{k\in\mathcal{D}_i}\mathbf{x}_k \tag{1.56}$$

we can calculate within the matrix

$$\mathbf{S}_W = \sum_{i=1}^{N}\mathbf{S}_i \geq 0 \tag{1.57}$$

where

$$\mathbf{S}_i = \sum_{k\in\mathcal{D}_i}(\mathbf{x}_k - \mathbf{t}_i)(\mathbf{x}_k - \mathbf{t}_i)^T. \tag{1.58}$$

The total and between matrices are calculated as

$$\mathbf{S}_{\mathrm{T}} = \sum_{k=1}^{m} (\mathbf{x}_k - \mathbf{x}_0)(\mathbf{x}_k - \mathbf{x}_0)^{\mathrm{T}}, \tag{1.59}$$

$$\mathbf{S}_{\mathrm{B}} = \mathbf{S}_{\mathrm{T}} - \mathbf{S}_{\mathrm{W}} = \sum_{i=1}^{N} m_i (\mathbf{t}_i - \mathbf{x}_0)(\mathbf{t}_i - \mathbf{x}_0)^{\mathrm{T}}. \tag{1.60}$$

When the pattern set is non-degenerated then $\mathbf{S}_{\mathrm{W}} > 0$ and we solve the generalized EVD problem, which is driven by equation

$$\left(\mathbf{S}_{\mathrm{W}}^{-1}\mathbf{S}_{\mathrm{B}} - \lambda \mathbb{I}_n\right)\mathbf{e} = \mathbf{0}. \tag{1.61}$$

The solutions of the generalized EVD can be ordered as $\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(N-1)} \geq 0$ with the corresponding eigenvectors $\mathbf{e}_{(1)}, \mathbf{e}_{(2)}, ..., \mathbf{e}_{(N-1)}$. Finally, the transformation matrix of the RAO method is

$$\mathbf{W}_3 = (\mathbf{e}_{(1)}, \mathbf{e}_{(2)}, ..., \mathbf{e}_{(N-1)}) \tag{1.62}$$

and the pattern set has a new feature matrix $\mathbf{Z} = \mathbf{X}_{\mathrm{S}}\mathbf{W}_3$ again.

The RAO method is well-informed and the new dimension is fixed to $N-1$, while the data whitening has the optional dimension of result and information about class membership is missing. Therefore, these two approaches can generate different matrices $\mathbf{W}$ and $\mathbf{Z}$. The RAO method is also useful for the data preprocessing.

1.5 Learning Quality

1.5.1 Quality Measures

The accuracy criterion is used as the traditional measure of a given classifier and can be expressed as a share of patterns qualified well. Using the concept of the class sensitivity as a relative frequency of the true classification, we can evaluate the accuracy for each class. After that, we can use the average sensitivity as a quality measure.

We prefer the critical sensitivity as the strength criterion of classifier efficiency which can be maximized via the union of hidden classes. We suppose the parameters of classification are set for the complete pattern set to obtain the classifier with maximum possible critical sensitivity se^*.

1.5.2 Cross-Validation Strategies

In the case of testing the classifier using all training data, we talk about naive validation.

After the learning we can perform the cross-validation strategies. When the number of patterns M is small, we prefer the Leave-One-Out [47, 48] cross-validation technique, but for a large M we can use the 10-fold [49, 50] cross-validation scheme, as generally recommended.

1.5.3 SOM Quality Measures

There are two main problems in the SOM learning. First of them is called a butterfly structure, when the patterns are mapped in the SOM graph with higher topographic error. The second problem is in the low accuracy of the self-organization when the weights of the SOM are far from the pattern set and the quantization error is higher. We will specify these measures first. The basic way of the quality measurement design is based on measuring the distances. The Euclidean distance of points $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^n$ is denoted $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$.

Using the pattern $\mathbf{x}_j$, we can investigate the distances to weights $\mathbf{w}_k$ and define winner as

$$\mathrm{win}(j) \in \underset{k=1,\ldots,H}{\arg\min}\, \mathrm{d}(\mathbf{x}_j, \mathbf{w}_k) \tag{1.63}$$

but the function win(j) is stochastic nature due to the possible distance equities. In some cases we did not find only one winner and the second winner is defined as

$$\mathrm{win2}(j) \in \underset{k \in \mathcal{M}_j}{\arg\min}\, \mathrm{d}(\mathbf{x}_j, \mathbf{w}_k), \tag{1.64}$$

where $\mathcal{M}_j = \{1, \ldots, H\} \setminus \{\mathrm{win}(j)\}$.

Using the distances and the winners, we can design traditional measures of various nature.

Distance Penalisation

The Quantization Error (QE) is traditionally related to all the forms of vector quantization and clustering algorithms [51]. Using linear penalisation we directly penalise the distances between the patterns and the corresponding winner weights as

$$QE_1 = \sum_{j=1}^{m} \mathrm{d}(\mathbf{x}_j, \mathbf{w}_{\mathrm{win}(j)}). \tag{1.65}$$

The quadratic penalisation

$$QE_2 = \sum_{j=1}^{m} \mathrm{d}^2(\mathbf{x}_j, \mathbf{w}_{\mathrm{win}(j)}) \tag{1.66}$$

is also used frequently but it has higher sensitivity to outliers.

Topographic Error

The general topographic rule is: if two objects are close to one another in reality they must be close to one another also in the map. Using this principle, the topographic error (TE) [52] is defined as

$$TE = 1 - \frac{1}{m}\sum_{j=1}^{m} g_{\mathrm{win}(j),\mathrm{win2}(j)}, \tag{1.67}$$

where $\mathbb{G} \in \{0,1\}^{H\times H}$ is the SOM topology matrix with $g_{u,v} = \mathrm{I}(\|\mathbf{p}_v - \mathbf{p}_v\|_2 \leq 1)$. The main advantage of the TE is its robustness to outliers.

Correlation based measures

The correlations between the mutual distances of patterns and the mutual distances of winner weights can be directly used as quality measures.

Let i, j be the pattern indices. The mutual pattern distances can be defined as $d_{i,j} = \mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$. The mutual distances of the corresponding weights are $\Delta_{i,j} = \mathrm{d}^*(\mathbf{w}_{win(i)}, \mathbf{w}_{win(j)})$, where d, d* are the corresponding distances.

Finally, we obtain $m(m-1)/2$ pairs of corresponding distances and directly calculate the Pearson correlation coefficient r, the Spearmann ρ, or the Kendall τ coefficient as a quality measure. The correlation coefficients are frequently declared as p–values of independence hypothesis H_0 to be comparable with the significant level 0.05.

Time Complexity of Measures

The evaluations of QE_1, QE_2 and TE are very fast with the time complexity $\mathrm{O}(mnH)$. The evaluation of correlation measures is more complex. The Pearson r has the time complexity $\mathrm{O}(mnH + m^2)$ due to simple statistics over $m(m-1)/2$ distance pairs. The Spearmann ρ is complicated by the pair sorting and its time complexity is $\mathrm{O}(mnH + m^2\log(m))$. The Kendall τ is not recommended for large pattern sets due to the time complexity $\mathrm{O}(mnH + m^4)$.

Composed Quality Measures

In our research, we prefer the QE_1 as the main optimal criterion. Due to its sensitivity to outliers, we propose using of QE_2 only as supplementary. Due to the higher time sentiment, we do not apply the correlation measures. The TE can be interpreted as a probability of topology saving in a random graph. Comparing TE as p-value with a critical level of α we can test the hypothesis H_0 that the resulting topology is random. Therefore, $TE \leq \alpha$ indicates a significant topology of the SOM mapping. Accepting only the significant topology, we can propose only QE_1 to avoid the butterfly effect.

Chapter 2

Thesis Goals

In this chapter, we summarize the main goals of the thesis. We focus on the sensitivity of the classification results instead of the commonly used accuracy. Our goal is not only to evaluate and compare the average sensitivity of classes but also to deal with the individual class sensitivity, i.e. to rely on the so-called critical sensitivity. Thanks to this approach, we expect to ensure better performance of the classifier and to achieve a more realistic evaluation of the classifier. The proposed approach avoids the situation when some of the classes having a small number of patterns would have a very small sensitivity while achieving a relatively high average sensitivity.

The main goal of this thesis is to propose the construction of a novel classifier based on a new principle. We propose the critical sensitivity representing the lowest sensitivity achieved in individual classes, playing the key role when building and evaluating the performance of the classifier. This means that our effort is to construct a classifier that ensures the achievement of quality for each individual class and will not omit the quality of underrepresented classes.

Another goal is to create the concept of the hidden classes, i.e. generally more classes than the output classes. Our expectation is that such classes could somehow help the classifier provide better results. The creation of these multiple hidden classes aims to improve the classifier to make the right decision in the required classification into a certain number of classes.

The next goal is to propose an imperfect learning strategy that would be based on the principle of the pattern set sub-sampling, the traditional classification, and dealing with the hidden classes concept. Such sub-sampling will subsequently lead to an increase in the quality of the classifier using the concept of class unification.

Another partial goal is to develop at least one new technique for the data self-organization that could serve as an alternative to the traditional Kohonen learning. We will therefore focus on the classical Kohonen learning and propose a new method for the adaptation of the learning weights. Our inspiration in this area will be the natural processes that are already mathematically describable and applicable to the process of the self-organization of data.

The next goal is to compare the effectiveness of individual heuristics with the traditional approaches using the standard cross-validation, as this approach will provide an objective comparison of the classifier quality. For this purpose, we plan to use several traditional and commonly available datasets for the classification.

Last but not least, we would like to apply the new classification technique to a real dataset and present the relevant result of the new classifier, including comparisons with the traditional solutions.

Chapter 3

Diffusion Based Learning

This chapter is focused on the novel approach to learning algorithms of self-organising maps. The aim is to benefit from a natural process of diffusion and present an alternative approach to the traditional Kohonen learning. Firstly we introduce general description of diffusion. Then we present traditional diffusion learning which is followed by anomalous diffusion.

3.1 Free Diffusion in $\mathbb{R}^d$

Supposing the diffusion of nitric oxide is the main slow learning phenomenon of neural systems, we have to model this process in a physical, chemical, and mathematical sense. We decided to analyse only such models of diffusion with the chemical reaction which offer an analytical solution in the infinite continuum of the given dimension. There are only two cases that satisfy the previous condition: traditional diffusion for exponent $\alpha = 2$ and anomalous diffusion constrained to the case when $\alpha = 1$.

The pudding model description begins with remembering basic facts. Let $m, n, H \in \mathbb{N}$ be the number of patterns, pattern dimensionality and a number of the SOM neurons [53]. The individual patterns are $\mathbf{x}_j \in \mathbb{R}^n$, where $j = 1, ..., m$ and they form the pattern set $\mathcal{S} = \{\mathbf{x}_1, ..., \mathbf{x}_m\}$. The fixed positions of individual neurons in the continuum are $\mathbf{p}_i \in \mathbb{R}^N$ for $i = 1, ..., H$ and they reflect the topology of the SOM, which is the subject of network design [54].

The diffusion process in the continuum can be easily expressed using matrix $\mathbf{D} \in (\mathbb{R}_0^+)^{H \times H}$ of distances $d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$. These mutual distances indirectly express the topology of the SOM. In the pudding SOM the neuron distances are not constrained to integers, which enables better space mapping. Therefore, the resulting SOM is invariant to the transition and rotation of its structure. Let $\Delta t > 0$ be the learning period and the diffusion in continuum will be studied only in discrete time $t_k = k \cdot \Delta t$, where $k \in \mathbb{N}_0$. The result of the SOM learning is the system of weights [55] $\mathbf{w}_i \in \mathbb{R}^n$, where $i = 1, ..., H$ of course. We begin with random weights setting $\mathbf{w}_i(0)$. The weights evolve during the learning process and their

values in time t_q are denoted as $\mathbf{w}_i(q)$, where $q \in \mathbb{N}_0$. The pudding model is based on substrate concentrations in neurons and given time. Being prepared for the SOM learning, we have to study the concentration profile first using single and complete activation procedures.

The case of traditional diffusion ($\alpha = 2$) is discussed in [56] and the main results introduce us to the formalism of the diffusion process.

3.2 Traditional Approach

The slow information transfer in the nervous system can be modelled as a diffusion process [57] with first-order chemical reaction [58]. The reactant is generated by single neuron activity [59] and the diffusion process [60] spreads the substance in the neuron neighbourhood. Our model is based on the second Fick's law [61] of diffusion, which is modified by the kinetics of pseudo-monomolecular [62] chemical reaction. The neuron activity can be modelled as a Dirac impulse in a given time. The main advantage of these simplifications is the existence of an analytical solution, which can be obtained as follows.

Let $N \in \mathbb{N}$ be space dimension, $\mathbf{y} \in \mathbb{R}^N$ be point coordinate, $D, t, \lambda > 0$ be the diffusion coefficient, time and the rate constant of a chemical reaction. The free diffusion of reacting substrate of concentration $\mathrm{c} : \mathbb{R}^N \to \mathbb{R}_0^+$ is driven by partial differential equation

$$\frac{\partial \mathrm{c}(\mathbf{y}, t)}{\partial t} = D \nabla^2 \mathrm{c}(\mathbf{y}, t) - \lambda \mathrm{c}(\mathbf{y}, t) \tag{3.1}$$

with initial condition

$$\mathrm{c}(\mathbf{y}, 0_+) = \delta(\mathbf{y}), \tag{3.2}$$

where $\delta : \mathbb{R}^N \to \mathbb{R}_0^+$ is the Dirac function.

The fundamental solution of (3.1) is

$$\mathrm{c}(\mathbf{y}, t) = \frac{1}{(4\pi D t)^{N/2}} \cdot \exp\left(-\frac{\|\mathbf{y}\|_2^2}{4Dt}\right) \cdot \exp(-\lambda t). \tag{3.3}$$

Due to system linearity, time and space invariance of (3.1), we can use the fundamental solution to study the multi-neuron system with sequential activities.

To be prepared for the SOM learning, we have to study the concentration profile first.

Single Activation

The pudding SOM learning is based on the activation of a single neuron. We will study the j-th neuron, which is supposed to be active in time t_k. Therefore, formally $j = \varphi_k$. But it is not necessary to study the substrate concentration at any

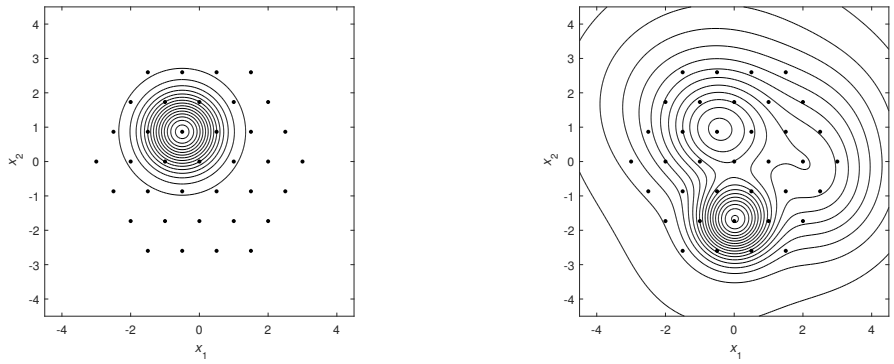


Figure 3.1: Concentration profile for normal diffusive learning after single (left) and complete (right) activation ($N = 2, a = 1, b = 1/10, H = 37, q = 100$)

point. The learning is based only on the concentration (3.3) in neuron points. The concentration in time t_q is

$$\mathrm{c}(\mathbf{y}, \mathbf{p}_j, t_q) = \frac{1}{(4\pi D(t_q - t_k))^{N/2}} \cdot \exp\left(-\frac{\|\mathbf{y} - \mathbf{p}_j\|_2^2}{4D(t_q - t_k)}\right) \cdot \exp\left(-\lambda(t_q - t_k)\right) \tag{3.4}$$

for $q > k$. The formula can be simplified to

$$\mathrm{c}(\mathbf{p}_i, \mathbf{p}_j, t_q) = \frac{1}{(4\pi D(q-k)\Delta t)^{N/2}} \cdot \exp\left(-\frac{d_{i,j}^2}{4D(q-k)\Delta t}\right) \cdot \exp\left(-\lambda(q-k)\Delta t\right). \tag{3.5}$$

After the substitution $a = 4D\Delta t > 0$, $b = \lambda\Delta t > 0$, we obtain resulting activation formula

$$\mathrm{c}(\mathbf{p}_i, \mathbf{p}_j, t_q) = (\pi a(q-k))^{-N/2} \cdot \exp\left(-\frac{d_{i,j}^2}{a(q-k)} - b(q-k)\right). \tag{3.6}$$

When $\min(d_{i,j} \geq 1)$, we suggest using $a = 1, b = 1/10$ for the first experiments, as it will be demonstrated in the next sections.

Complete Activation

The SOM learning is based on substrate concentrations in the q-th step in time t_q. This concentration is the result of previous activation sequence $\varphi_1, \varphi_2, \ldots \varphi_{q-1}$ using single activation model (3.6). Due to the linearity of (3.1) we can use the additivity principle and directly calculate the cumulative concentration in the i-th neuron and step q as

$$c_{i,q} = \sum_{k=1}^{q-1} c(\mathbf{p}_i, \mathbf{p}_{\varphi_k}, t_q - t_k) = \frac{1}{(\pi a)^{N/2}} \cdot \sum_{k=1}^{q-1} \frac{\exp\left(-\frac{d_{i,\varphi_k}^2}{a(q-k)} - b(q-k)\right)}{(q-k)^{N/2}}. \tag{3.7}$$

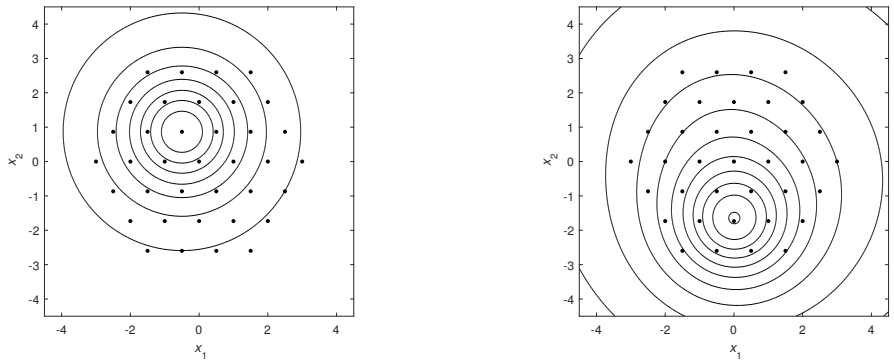


Figure 3.2: Concentration profile for anomalous diffusive learning after single (left) and complete (right) activation ($N = 2, a = 1, b = 1/10, H = 37, q = 100$)

The resulting formula consists of all concentration information that is necessary for the SOM learning. Therefore, the concentration $c_{i,q}$ is only a function of activation history, the SOM topology, and parameters a, b. The difference between single and complete activation is depicted in Figure 3.1 for normal diffusive learning. The concentration of substrate is very high in the neighbourhood of the last winning neuron in history, which is a result of learning.

3.3 Anomalous Diffusion

As rarely observed in nature, the anomalous diffusion [63] is a more complex alternative to the traditional one. Both formulation and solution of models with anomalous diffusion are very complicated and not trivial, except for the case when $\alpha = 1$, which is sometimes called ballistic diffusion. We will formulate the model in a general form first. Let $1 \leq \alpha < 2$, $D_\alpha > 0$ be anomalous exponent and diffusion coefficient, respectively. The free anomalous diffusion of reacting substrate of concentration $\mathrm{c} : \mathbb{R}^N \to \mathbb{R}_0^+$ is driven by partial differential equation

$$\frac{\partial \mathrm{c}(\mathbf{y}, t)}{\partial t} = D_\alpha \, \nabla^{(\alpha)} \, \mathrm{c}(\mathbf{y}, t) - \lambda \mathrm{c}(\mathbf{y}, t) \tag{3.8}$$

with initial condition

$$\mathrm{c}(\mathbf{y}, 0_+) = \delta(\mathbf{y}), \tag{3.9}$$

where $\delta : \mathbb{R}^N \to \mathbb{R}_0^+$ is the Dirac function.

The explicit solution is obtainable only for $\alpha = 1$. The fundamental solution for $\lambda = 0$ is probability distribution function of multi-varietal Cauchy distribution [64] for scale $\gamma = D_1 t$

$$\mathrm{c}(\mathbf{y}, t) = \frac{\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}} \cdot \frac{D_1 t}{(D_1^2 t^2 + \|\mathbf{y}\|_2^2)^{(N+1)/2}}. \tag{3.10}$$

Using shift theorem [65] of Laplace transform, we obtain the general solution

$$\mathrm{c}(\mathbf{y},t)=\frac{\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}}\cdot\frac{D_1 t\cdot\exp(-\lambda t)}{(D_1^2t^2+\|\mathbf{y}\|_2^2)^{(N+1)/2}} \tag{3.11}$$

and therefore,

$$\mathrm{c}(\mathbf{y},\mathbf{p},t_q)=\frac{\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}}\cdot\frac{D_1(t_q-t_k)\cdot\exp(-\lambda(t_q-t_k))}{(D_1^2(t_q-t_k)^2+\|\mathbf{y}-\mathbf{p}\|_2^2)^{(N+1)/2}}, \tag{3.12}$$

$$\mathrm{c}(\mathbf{p}_i,\mathbf{p}_j,t_q)=\frac{\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}}\cdot\frac{D_1(q-k)\Delta t\cdot\exp(-\lambda(q-k)\Delta t)}{(D_1^2(q-k)^2(\Delta t)^2+d_{i,j}^2)^{(N+1)/2}}. \tag{3.13}$$

After the substitution $a=D_1\Delta t>0$, $b=\lambda\cdot\Delta t>0$, we obtain

$$\mathrm{c}(\mathbf{p}_i,\mathbf{p}_j,t_q)=\frac{a\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}}\cdot\frac{(q-k)\cdot\exp(-b(q-k))}{(a^2(q-k)^2+d_{i,j}^2)^{(N+1)/2}}, \tag{3.14}$$

$$\mathrm{c}_{i,q}=\sum_{k=1}^{q-1}\frac{a\Gamma(\frac{N+1}{2})}{\Gamma(1/2)\pi^{N/2}}\cdot\sum_{k=1}^{q-1}\frac{(q-k)\cdot\exp(-b(q-k))}{(a^2(q-k)^2+d_{i,j}^2)^{(N+1)/2}}. \tag{3.15}$$

The learning efficiency depends on concentration profiles. Therefore, it is useful to compare substrate concentrations in the case of anomalous diffusion with the normal case. Results of single and complete activation are depicted in Figure 3.2 for anomalous diffusive learning. As seen in Figures 3.1 and 3.2, the substrate is more spread out in the case of anomalous diffusion and the difference between minimal and maximal concentration is smaller. These differences between normal and anomalous diffusions influence the SOM learning algorithm. The anomalous and traditional diffusion modelling in more detail is discussed in [66].

Chapter 4

Theory of Imperfect Parallel Learning

In this chapter, we present the general theory and origins of the imperfect learning. The aim is to underline the motivation for the proposed approach. We summarize basic methods of pattern sub-sampling, feature sub-samplings as well as the use of imperfect learning algorithms. The following part is focused on the novel approach based on the hidden classes. Firstly, we provide a basic framework regarding the forming of hidden classes. In the last part, we focus on the process of optimal union of hidden classes which is focused on the critical sensitivity of final classification and present the related theory.

4.1 Origins of Imperfect Classification

When the classifier learning process is driven by standard perfect algorithms, we have no guarantee to obtain the requested properties, e.g. perfectness, accuracy, or sensitivity. This poor quality can be caused by several reasons. One of them is a non-separable pattern set. Several learning algorithms find only the local extreme instead of the global one. To prevent such obstacles, we reach the result in an alternative way, where we do not insist on the algorithm perfectness. This approach can lead to better performance of classification tasks even when there is no perfect solution.

Thanks to the stochastic nature of the imperfect learning, we can apply such algorithm repeatedly NMC times on the pattern set $\mathcal{S}$ and obtain desired results. The evaluation results is discussed in Section 4.3. Therefore, we appreciate all the imperfect learning results and we store them. Our aim is not to select the best one but to work further with all the different results of imperfect classifiers providing different results on the same pattern set. There can be any kind of reason for the imperfectness.

One way of the imperfect learning can be represented by selecting only a limited number of patterns and not the whole pattern set, which is somehow a context-out approach. Using different subsets of pattern set leads to different classifier results. The second way is to focus only on some features, which are selected randomly and represent the context-out technique again. Both ways can be combined as well. The third way is to use any known imperfect method which guarantees to find only a local extreme. Instead of being stressed by the poorly working original method, our method yields from it.

4.1.1 Pattern Sub-sampling

Having m patterns, we can randomly select m^+ patterns, where $m^+ \geq m$, and try the learning of the system by using the potentially incomplete data. Therefore, the learning is not perfect and the resulting classifier parameters can differ from the parameters using a complete data set. Using the set $\mathcal{I}^* = \{1, ..., m\}$ of pattern indices, we will define index set

$$\mathcal{I}_k \subset \mathcal{I}^* = \{1, ..., m\}, \tag{4.1}$$

which is generated recursively starting with $\mathcal{I}_0 = \{\}$, generating $i_k \sim \mathrm{U}(\mathcal{I}^* \setminus \mathcal{I}_{k-1})$, and updating $\mathcal{I}_k = \mathcal{I}_{k-1} \cup \{i_k\}$.

Finally, we create the training pattern set as

$$\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i^*) : i \in \mathcal{I}_{m^+} \subset \mathcal{S}\} \tag{4.2}$$

as a randomly generated subset of size m^+. But when $m^+ = m$, we directly use $\mathcal{T} = \mathcal{S}$.

4.1.2 Feature Sub-sampling

We can also randomly reduce the feature of the patterns. Therefore, we define $\mathcal{J}^* = \{1, ..., n\}$ and select only n^+ features, where $n^+ \leq n$. The index set

$$\mathcal{J}_k \subset \mathcal{J}^* = \{1, ..., n\} \tag{4.3}$$

is generated recursively starting with $\mathcal{J}_0 = \{\}$, generating $j_k \sim \mathrm{U}(\mathcal{J}^* \setminus \mathcal{J}_{k-1})$, and updating $\mathcal{J}_k = \mathcal{J}_{k-1} \mathrm{U} \{j_k\}$.

Using the reduced vector $\mathbf{r}$ of n^+ features, we can define the reduced pattern

$$\mathbf{p}_{red} = (\mathbf{r}, \mathbf{y}^*), \tag{4.4}$$

where $\mathbf{r} \in \mathbb{R}^{n^+}$. When $n^+ = n$, we directly set $\mathbf{r} = \mathbf{x}$. Finally, the reduced patterns form the training pattern set.

4.1.3 Imperfect Learning Algorithm

The last presented way of the imperfect learning is using an algorithm of stochastic nature. The repeated use of such algorithm leads to different results and whether the provided result is the best one is irrelevant. The stochastic nature can be formed by a stochastic estimate of the initial parameter value followed by a deterministic approach. A typical example of such algorithm is the clustering by the K-means technique. Another way, independent of the way of forming the initial parameter, is the stochastic nature of a step in the iterative learning methods. The typical way is represented by methods which use one of the randomly selected patterns. There are many methods of such kind, e.g. perceptron learning, multilayer perceptron, the SOM or the RBF. All these methods are well known in the literature, very often we speak about stochastic gradient methods or local minimizers. The best-known methods are represented by Hebb [67], Rosenblatt [68], Widrow & Hoff [69], and Kohonen learning [34].

The main advantage of the repeated imperfect learning, whether on the complete or the reduced pattern set, is that the stored and analysed results lead to a classier with better properties.

4.2 Hidden Classes Forming

After performing random experiments with the classification in the sense of the sub-sampling in pattern feature space or using the stochastic techniques of learning, we obtain various results of the classification. We denote $M, M^*, H \in \mathbb{N}$ number of involved classifiers, reduced number of classifiers, and the resulting number of hidden classes. Having M classifiers $c_k : \mathbb{R}^n \to \{0, 1, ..., N\}$ for $k = 1, ..., M$, we can apply them to the original pattern set and the result can be collected in matrix $\mathbf{Y}$, $\mathbf{Y} \in \{0, 1, ..., N\}^{m \times M}$ with elements $y_{i,j} = c_j(\mathbf{x}_i)$.

Although each classifier is different, they can lead to the same results for given pattern set. Therefore, it is necessary to reduce the number of columns first by eliminating the duplicities. Resulting reduced matrix is $\mathbf{U} \in \{0, 1, ..., N\}^{m \times M^*}$ consists of M^* unique responses, where $M^* \leq M$.

The hidden class $\mathscr{H}_j$ is defined as a set of patterns that produce the same row in matrix $\mathbf{U}$. Therefore, it is necessary to reduce the rows of $\mathbf{U}$ to the unique ones, which are collected in matrix $\mathbf{P}$, $\mathbf{P} \in \{0, 1, ..., N\}^{H \times M^*}$. The matrix $\mathbf{P}$ is a condensed summary of the classification results. Using the Hamming distance $\|...\|_{\mathrm{H}}$, we can decide whether $\mathbf{x} \in \mathbf{H}_j$ by verification that

$$\|\mathbf{u} - \mathbf{p}_j\|_{\mathrm{H}} < \|\mathbf{u} - \mathbf{p}_k\|_{\mathrm{H}} \tag{4.5}$$

for all $k \neq j$, where $\mathbf{u} \in \{0, 1, ..., N\}^{M*}$ is the reduced response to $\mathbf{x}$. In any cases, the hidden class is unknown and labelled by zero. The result of partitioning into hidden classes is a hidden class number $j^* \in \{0, ..., H\}$.

4.3 Union of Hidden Classes

In the case of the classification, we are well motivated to define the final (output) classes because of their close connection to the solution of the problem. The cardinal question is why to define and use other classes called hidden classes. There is a good analogy with ore mining. The formulation of the mining problem is clear. It is necessary to separate the material into two classes: the ore and the residual material. But for a large stone, the task is too complex. First, it is necessary to break it into small pieces as symbols of the hidden classes and then carefully sort them into two output classes: the ore and the rest by using an effective procedure. Although the technical aspects of the separation are also useful, they are not discussed here. We will focus on the decision whether a given piece of stone is ore or not. Using the majority rule, the pieces with more than fifty percent of ore belong to the first class (ore). It is a traditional approach but with very low efficiency in general. For example, the concentration of gold is very low, and therefore no piece of stone would be selected for future gold extraction. This paradox can be easily solved by using class sensitivity and critical sensitivity, which help to construct more sophisticated strategies not only for ore mining but mainly for the general classification of patterns.

A general classification task distributes m patterns into N classes, and our method is based on the preprocessing which places them into H hidden classes. The main idea of this is to break down a complex task into more simple ones, where we will enable its fast solution using very fast algorithms. One of the algorithms that are directly offered for this purpose is the cluster analysis. The formation of hidden classes is very intuitive in this case. For such purpose, we focus on self-organizing maps and present an innovative approach based on the chemical reaction.

There are many approaches to the performance of hidden classification using various kinds of local classifiers, and they are generally imperfect. Any hierarchical classifier consists of one hidden layer at least. There are many possibilities for designing such a hidden layer. We prefer the parallel system of imperfect classifiers. The imperfect classifier is any system that classifies the patterns to given output classes but with low efficiency. Having many classifiers of such kind, we can use them for alternative pattern description and forming of the hidden classes. As in real life, we can focus only on several cases and particular features. The imperfect classifier can be learned in any traditional way of the perfect classification but only using several patterns and several properties. This approach is called context out learning here.

After forming the hidden classes, we face a major challenge to form the final classes. The aim is to find an optimal way to unite these hidden classes. We search optimal method to put together the hidden classes so that they can form the final classes for classification. Our approach is based on the basic characteristics of classification quality, which are frequently used in many applications: accuracy, class sensitivity, and critical sensitivity.

The presented novel approach of vector pattern multi-classification is based on the combination of the approaches mentioned above. Basic assumptions are the following. Let N, M, n be the number of output classes, the number of patterns, and the number of pattern dimensions that are unlimited in general. But there is a threshold value n^* of pattern dimension, which switches between deterministic and random sub-sampling approaches. In both cases, the first step of classification is the dimensionality reduction using the data whitening or the multi-class discriminant analysis which converts the data into the space of dimension $D \leq n$. In the second step, the reduced data are clustered using the DBSCAN technique and the hidden classes are formed. Optimal unions of these hidden classes (OUHC) are performed in the last step of the multiple classifications.

Presenting a new way of the classification requires sufficient testing. Our aim is to present the result of the new classifier on standard classification tasks and present the comparison with standard methods. For this purpose we use the iris dataset.

Our aim is to present how an imperfect procedure can bring better results than the application of sophisticated and at the same time computationally very demanding alternatives. The imperfect approaches can be represented by the sub-sampling. The initial phase is usually the data preprocessing.

Both supervised and unsupervised approaches to the pattern classification can be used to form hidden classes inside the final classifier. The system of hidden classes arises from the uncertainty of class membership, the imperfectness of classification, or any context-out approach. We will apply a deterministic approach which is based on the relationship between the hidden groups and the output classes [70] as follows.

The aim is to optimize this relationship as the best mapping from the hidden to the output classes. The strict classifier is defined as mapping $\mathrm{c} : \mathscr{L}_H \to \mathscr{L}_N$ from the set $\mathscr{L}_H$ of hidden class indices to the set $\mathscr{L}_N$ of final class indices, where $\mathscr{L}_n = \{1, ..., n\}$. This mapping can be expressed via the matrix $\mathbb{X} \in \{0,1\}^{N \times H}$, where $x_{i,j} = 1$ if $d_k \in \mathscr{H}_j \Rightarrow d_k \in \mathscr{C}_i$. Therefore, $x_{i,j} = 1$ just when for any pattern belonging to $\mathscr{H}_j$ it also belongs to $\mathscr{C}_i$. The uniqueness conditions $\sum_{i=1}^{N} x_{i,j} = 1$ have to be satisfied for $j = 1, ..., H$.

The relation between the classes and the hidden groups is presented via the contingency table $\mathbb{F} \in \mathbb{N}_0^{N \times H}$, where $f_{i,j} = \mathrm{card}\{k : d_k \in \mathscr{C}_i \bigcap \mathscr{H}_j\}$ is the result of the pattern counting. Here, $f_{i,j}$ is the number of patterns belonging to both class $\mathscr{C}_i$ and group $\mathscr{H}_j$ as joint frequency, which can be relativized as

$$q_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{H} f_{i,k}}, \tag{4.6}$$

where $i = 1, ..., N$, $j = 1, ..., H$.

The accuracy of a given classifier can be expressed as

$$acc = \frac{1}{m} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j}. \tag{4.7}$$

Using the concept of the class sensitivity as a relative frequency of the true classification, we can calculate it for $i = 1, ..., N$ as

$$se_i = \sum_{j=1}^{H} q_{i,j} x_{i,j}. \tag{4.8}$$

An average sensitivity can be defined as

$$ase = \frac{1}{N} \sum_{i=1}^{N} se_i. \tag{4.9}$$

A lower estimate of the class sensitivity is defined as a critical sensitivity

$$se^* = \min\{se_i : i = 1, ..., N\}. \tag{4.10}$$

We prefer the critical sensitivity as the strength criterion of the classifier efficiency and maximize it via the union of hidden classes. The accuracy criterion is also used as the traditional measure by many authors.

In accordance with [70], we will maximize the critical sensitivity se^*. An adequate mixed binary optimization task is

$$se^* = \max \tag{4.11}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \text{ for } j = 1, ..., H, \tag{4.12}$$

$$\sum_{j=1}^{H} q_{i,j} x_{i,j} - se^* \geq 0 \text{ for } i = 1, ..., N, \tag{4.13}$$

$$x_{i,j} \in \{0, 1\} \text{ for } i = 1, ..., N,\ j = 1, ..., H, \tag{4.14}$$

$$se^* \in [0, 1] \tag{4.15}$$

with real artificial variable se^*. The inequalities (4.13) guarantee that se^* is the lower bound of the critical sensitivity during the optimization process.

After the specification of se^*, we can yield from the task degeneration and solve additional binary programming task, which guarantees the same critical sensitivity and maximizes accuracy as

$$acc = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j} = \max \tag{4.16}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \text{ for } j = 1,...,H, \tag{4.17}$$

$$\sum_{j=1}^{H} q_{i,j} x_{i,j} \geq se^* \text{ for } i = 1,...,N, \tag{4.18}$$

$$x_{i,j} \in \{0,1\} \text{ for } i = 1,...,N,\ j = 1,...,H. \tag{4.19}$$

Chapter 5

Imperfect Classifier

We summarize the results from the previous theoretical parts in this chapter, which means that we obtain the classifier description. Firstly we demonstrate the structure of the proposed classifier. We discuss the applicable data transformations and classification methods. The crucial part is represented by hidden classification followed by a summary of validation methods.

5.1 Classifier Structure

The proposed classifier design is captured in Figure 5.1. In this figure, we see q imperfect classifiers in the central part being at the same level of importance for us. It means that we consider all individual results with the same importance. The individual results represent the core of the proposed system. The layer of imperfect classifiers represents a hidden layer of our system.

In the proposed system, we use parallel classifiers after a previous preparation of input data. Before using the classifiers, we decide on an optional non-linear transformation, and subsequently, we can still perform an optimal linear transformation. The first transformation aims to make the dataset linearly separable. While the goal of the linear transformation is, for example, to lose as little data variance as possible when reducing the dimensions or reaching the separable classes. This approach enables us to obtain a lower dimension of the dataset and to maintain the variance. The choice of preprocessing methods is user-based, and we assume that all the classifiers in the third hidden layer work with the same dataset. Each classifier has its unique view of the dataset.

The next step of the proposed classifier is based on a hidden classifier. This hidden classifier creates the final vector description of the initial dataset. In the next step, the final classification processes OUHC to obtain the right representatives of classes. The learning of the final classification in general means obtaining the system parameters. Optional nonlinear transformation can be learned by specific methods, while linear transformation is usually learned by the traditional classifiers described in the next section.

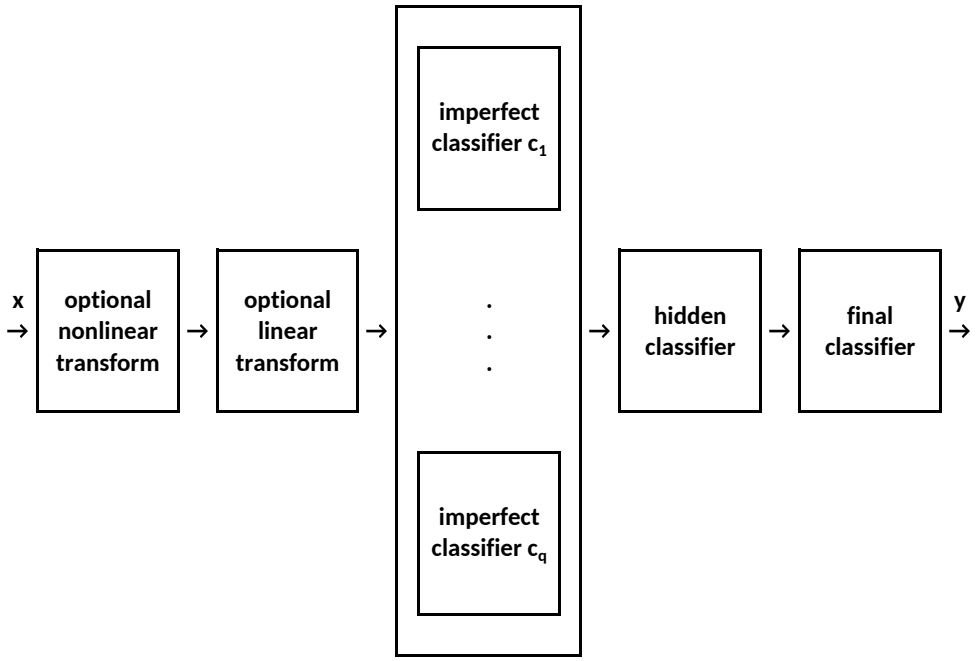


Figure 5.1: Classifier Structure

The imperfect classifiers can be learned in any way, whether deterministic or stochastic, on a training set. It should be noted that it is typical that the non-perfect classifiers do not have to learn on the whole set by a non-perfect algorithm but they can learn by a perfect algorithm on parts of a dataset (4.2) or on parts of properties (4.4), which was described in Sections 4.1.1 and 4.1.2. It is important that, this way we achieve a variety of classifiers on the training set. If it is not the case, we suggest removing the redundant classifiers. Then it is important to realize that when we know the results of classifiers on an imperfect set and we have formed the hidden classes described in Section 4.3. These hidden classes can be unified using linear programming methods for critical sensitivity maximization (4.11-4.15) and additionally for accuracy maximization (4.16-4.19) to create the final classifier. Therefore, the result of learning is not only the set of imperfect classifiers but also the elimination of redundancy and the formation of certain patterns and subsequent optimal unification. The classifier itself then uses these learning outcomes to classify a particular object **x**. The system of proposed classifiers contains many possibilities to combine the classification approaches and their responses [71].

5.2 Applicable Non-Linear Transformations

When we decide to apply the non-linear transformations, we have the opportunity to use the following, commonly used methods. The first one is focused on non-negative values when we can use logarithmic transformation

$$y = \ln x, \text{for } x > 0. \tag{5.1}$$

The next possibility is to use Box-Cox transform [72]

$$y = \frac{x^{\alpha} - 1}{\alpha}, \text{for } x > 0, \alpha \neq 0. \tag{5.2}$$

In the case when $\alpha \to 0$, the Box-Cox transform (5.2) approaches the logarithmic one (5.1). These transformations are recommended for all positive features which differ significantly in order.

Another option applicable to all the variables is to use polynomial expansion, which means constructing a polynomial base in n-dimensional space of finite order. The advantage of using the polynomial expansion on centred data is that we usually end up with a second-order or third-order polynomials.

In addition, it makes sense to use any other transformations associated with a particular application for which there exist any technical or economic reasons. We can use the transformation only on selected variables or omit the non-linear transformation completely.

5.3 Applicable Linear Transformations

The goal of the linear transformations is to reach selected basic data properties. For example, zero mean values, a unit covariance matrix, a diagonal covariance matrix, or to obtain data with higher separability between classes.

We can perform linear transformation also without any knowledge of class affiliation. The main aim is to extract anything from the raw dataset. A method suitable for those purposes is the Principal Component Analysis (1.46), where we obtain a lower number of classes, a diagonal covariance matrix, and zero means. The application makes sense if we want to get a lower number of components than the size of the original data. If we also insist on certain standardization of data, then we should use the data whitening (1.52), which will ensure that the covariance matrix will be an identity one. In this case, it is up to us whether we keep the number of components or reduce it.

If we know the affiliation to the classes, we offer the use of multi-class discrimination analysis inspired by the RAO method (1.55 - 1.62), which creates weights differently to maximize the differences between the classes. The method generates a fixed number of coordinates that is one less than the number of classes.

5.4 Applicable Classification Methods

In the research part, we have presented a number of learning principles that can be used in classification tasks. All principles used in our proposed classifier are located in the central part as parallel classifiers in the sense of Section 5.1. The supervised learning techniques are summarized in the first part of Table 5.1. The unsupervised learning techniques are listed in the second part of Table 5.1.

Table 5.1: List of Classification Techniques

Abbreviation	Method	Section
k-NN	k-Nearest Neighbour [10]	1.2.1
RR	Ridge Regression [14]	1.2.2
KRR	Kernel Ridge Regression [14]	1.2.2
LDA	Linear Discriminant Analysis [3]	1.2.1
QDA	Quadratic Discriminant Analysis [4]	1.2.1
RVFL	Random Vector Functional Link [16]	1.2.3
MM	Max-Margin [11]	1.2.2
Parzen	Parzen Windows [5]	1.2.1
LQ	Loftsgaarden-Quesenberry [9]	1.2.1
K-means	K-means Clustering [10]	1.3.1
SOM	Self-Organizing Map [39]	1.3.2
SLINK	Single Linkage [23]	1.3.1
DBSCAN	Density-Based Spatial Clustering of Applications with Noise [10]	1.3.1

If we want to perceive the resulting classifier very optimistically, we can get the first optimistic (naive) idea of the quality of the classifier directly on the learning set. We will create a contingency table that will compare the actual class affiliations and the proposals of the resulting classifier. From this contingency table, we can easily obtain the accuracy (4.7), sensitivity (4.9), and critical sensitivity (4.10) of the classifier. Such an approach represents a very straight, very optimistic, and very misleading idea. Therefore, we will test the classifiers using the cross-validation technique.

5.5 Formation of Hidden Classes

After finishing the first three layers of the proposed classifier, which means following the optional transformation and learning the central part of the proposed classifier, we have to formulate what we call a hidden class. For each pattern, we obtain a group of results in a vector, which means forming a new set of patterns. The fundamental theory of this is described in Section 4.2.

The natural part of the formation of new patterns is, firstly, the removal of redundant duplicate columns, and secondly, the unification, which means the search for unique patterns that will become representatives of the hidden classes. The number of patterns thus determines the number of hidden classes. Therefore, it is clear to which hidden and final class the vector belongs. The process of OUHC is described in Section 4.3.

If we solve the network response to a specific vector, we must first undergo optional preprocessing and go through all the classifiers of the central layer of the proposed classifier, thus obtaining the one integer pattern that we either find

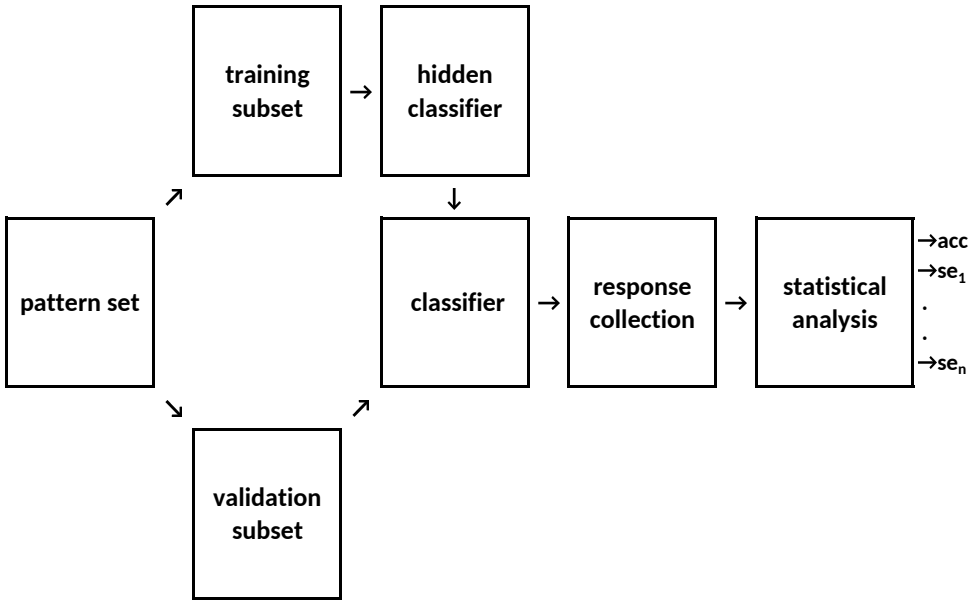


Figure 5.2: Simple Cross-Validation Scheme

immediately and know which class it belongs to or we do not find it, and we investigate which pattern it is closest to in terms of Hamming distance (4.5). If there is no decision after that, the classifier's response is that it does not know the class affiliation for the pattern.

5.6 Validation Methods

There are three approaches recommended for the validation. The first approach depicted in Figure 5.2 is used mainly for problems where the pattern sets arose over time, so we usually know the history and we can distinguish between ancient and recent history. In this case, the systems typically learn from the older history and verify against the recent history. This approach is thus typical for the classification systems used for time-series predictions across a variety of disciplines.

If we build predictors for time series, we have to create a training set from data within a certain time, so the first part of the patterns is used for the training and the remaining part for the verification. According to the Figure 5.2 we see that we are creating the hidden classes and a final classifier from the training subset. Then we let the rest of the data into the finished classifier and find out the classifier's responses. Finally, we perform the statistical analysis and construct the contingency tables.

If the pattern number has nothing to do with time, cause, and effect, then systematic or the random pattern selection should be preferred. If we have little data or a powerful computer technology, then the cross-validation method is offered. This method is depicted in Figure 5.3. The Leave-One-Out method, mentioned in Section 1.5.2, proceeds in such a way that the k-th pattern is used for validation,

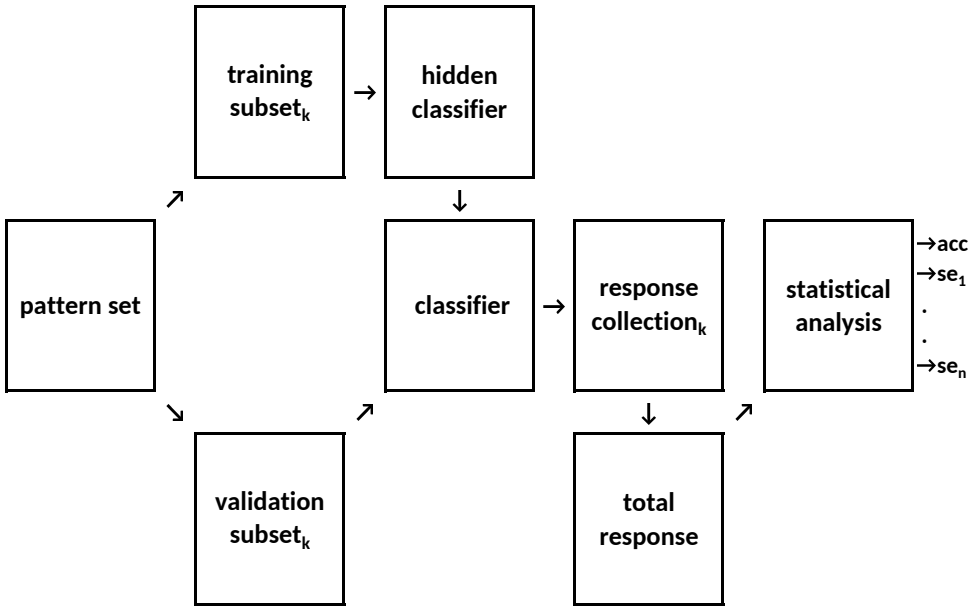


Figure 5.3: Advanced Cross-Validation Scheme

while the other patterns form a training set and we learn only one row of the contingency table. In contrast to Figure 5.2, we receive only one-row response in each step, therefore, we have to collect all responses as seen in Figure 5.3 when we collect all responses for $k = 1,...,m$. If we gradually exhaust all patterns in the cycle, it is a time-consuming process, but we learn the truth about the actual behaviour of the classifier.

The third validation method to be used is M-fold validation. This method represents a compromise validation solution between the naive and the Leave-One-Out approach. To ensure the fairness of this method, we first randomly shuffle the patterns and then divide them into groups of almost equal sizes. In the k-th step, the validation subset is the k-th group, and the remaining data are in the training set. In this case, the response collection consists of as many patterns as there are members in the M-fold. If we go through all the piles in the cycle, we get the overall response, which we analyse again. The 10-fold validation is most often used in the literature, but from the point of view of statistical stability, we recommend taking into account the total number of samples and the 50-fold validation has proved its worth. Therefore, the advanced cross-validation scheme of M-fold validation is formally the same (see Figure 5.3) but with $K = 1,...,M$, and every validation subset consists of approximately m/M patterns.

Chapter 6

Experimental Part

In this chapter, we demonstrate the properties of a proposed imperfect classifier. We follow previously defined approach and the aim is to present an independent comparison of new classifier results. In all presented results we use the data preprocessing in four basic forms. The first one is using the RAO method and the other three are represented by the one-, two-, and three-dimensional data whitening. The aim of the testing is discussed in more detail in the first part of this chapter.

The standard datasets have been used for the perfect learning and the proposed imperfect learning. Firstly, we focus on the iris dataset using 14 referential methods in detail. For each method, we provide the comparison of results for the perfect and the proposed imperfect learning together with the method's optimal parameter settings. The perfect learning means the method is used in a standard way for the original dataset together with the standard techniques of preprocessing. The imperfect learning means using the same method with the pattern and feature reduction together with the repeated learning.

In the next part, we present the selected methods for another ten standard datasets in order to present a general picture of the proposed classifier behaviour. The critical sensitivity as a benchmark remains to evaluated. The aim is to present and to compare the results following this classification approach.

In the last part of this chapter, we present a proposed classifier for the classification of the real dataset used for the leaves classification and crisis prediction. In the first case, the description of ten different trees is represented by the mathematical interpretation of their photographs taken under different conditions. The second task is based on the economic descriptors for different economies.

6.1 Aims of Testing

The testing objectives are as follows. We aim to find the basic rules of the imperfect classifier design on the given datasets and to compare them with the individual cases. It means if one type of the classifier used individually is better or worse

when used in the imperfect parallel mode. We aim to find out which classifiers are improved by this approach and which ones are not affected.

The first goal is to investigate whether the individual or the parallel use leads to better results related to the classifier type. If the classifier has low classification capabilities, for example, due to its linearity, then under what conditions will it be possible to improve it by the imperfect learning and under what conditions? The second question is whether the imperfect classifier over-performs the traditional one when efficient on a given dataset. The goal also is to investigate the optimal number of those imperfect classifiers and evaluate the level of imperfection leading to the best results. We also investigate whether the imperfectness principle or the classification technique is enough, it may not, for example, converge well, or whether it is more appropriate to do the imperfection by choosing patterns and properties.

6.2 Preludium: Iris Flower Classification

Firstly, we present the effect of imperfections by a method that is perfect in itself and has few parameters. The goal is to demonstrate whether we are able to improve the given method. We have selected a traditional classification task for this purpose, the IRIS dataset. We focus on fourteen different methods as stated in Table 6.1. The optimal settings for each method for the perfect learning as well as for the imperfect learning are included in mentioned table. The main purpose of the comparison is to present the power of the proposed method compared to the perfect ones.

Firstly, we demonstrate the results for all the classifiers and one dataset, the iris dataset, using the cross-validation. For all tasks, we investigate different preprocessing regarding the data dimension. It means we have used the RAO method as well as the data whitening for $D \in \{1, 2, 3\}$. The basic framework of proposed preprocessing is used in both the perfect and the imperfect learning.

We search for the best results for the same parameter settings in both cases. Therefore, the initial setting of the method's parameters is the same for the perfect and the imperfect learning. It means we use the same parameters in both cases to find the best results.

In the case of the imperfect learning, we focused also on the impact of the number of repeated preprocessing, selecting a different number of the patterns and the features. The experiments led us to the proposed reduction to 2/3 in the case of the number of patterns and a ten to twenty percent lower number of the features. Therefore, in the case of the iris dataset, the number of the features after reduction for the imperfect learning strategy is set as $m^* = 100$ and the number of features is set as $n^* = 3$. The sufficient number of repetitions is 100. The detailed comments on the results for individual methods are as follows.

The important results represented by the values of the critical sensitivity together with the parameter settings for concrete methods are summarized in Table 6.1.

This table shows the parameter setting for each classifier to obtain the best classification results. The detailed results for each method are captured in Figures 6.1 to 6.15. Each figure covers four preprocessing techniques - the RAO method preprocessing and the data whitening for the three-dimensionality reductions.

Using k-NN classifier directly we reach the critical sensitivity at 0.94, for example using the RAO method preprocessing and the parameter $k = 3$. Using the imperfect learning, we reach the same value of the critical sensitivity at 0.94, for example for the RAO method preprocessing and k=7. Therefore, we see that in the case of k-NN classifier, the imperfect learning brings the same results without any improvement. All results for the different values of k settings are captured in Figure 6.1.

The direct ridge regression reaches the highest critical sensitivity at 0.68 and it is constant for λ close to zero and the RAO method preprocessing. The imperfect learning leads to critical sensitivity at 0.82 for the parameter λ close to zero as well for both the RAO method preprocessing and three-dimensional data whitening. The classification results are significantly better for the imperfect learning techniques compared to the perfect ones. All the results for different settings of parameter λ are captured in Figure 6.2.

The presented classification based on the kernel ridge regression is devoted to the best set of the parameter λ in the previous case of the ridge regression. The highest value of the critical sensitivity for the perfect learning is 0.94. This value was reached for example for the RAO method preprocessing and the parameter $\sigma = 0.5$. The imperfect learning leads to the critical sensitivity at 0.96 for example for the two-dimensional whitening and $\sigma = 0.1$. The already high critical sensitivity in the case of the perfect learning strategy was increased to 0.96 using the imperfect learning. All the results for different settings of parameter σ are captured in Figure 6.3.

The direct classification using the LDA method reaches the critical sensitivity at 0.96, for example for the RAO method preprocessing and parameter $\lambda = 0.1$. When we used the imperfect learning strategy, we reached the critical sensitivity at 0.98, for example in the case of the two-dimensional data whitening and $\lambda = 0.01$. For the LDA, the high critical sensitivity was increased to 0.98 using an imperfect learning strategy. All the results for the different settings of parameter λ are captured in Figure 6.4.

The direct use of the QDA method leads to the critical sensitivity at 0.96, for example for the three-dimensional data whitening and the parameter $\lambda = 0.01$. When we used the imperfect learning strategy, we reached the critical sensitivity at 0.98, for example in the case of the three-dimensional data whitening and $\lambda = 0.01$. For the QDA there is the similar optimal setting for the perfect and the imperfect cases, but the imperfect learning reaches the higher critical sensitivity. All the results for the different settings of parameter λ are captured in Figure 6.5.

The highest critical sensitivity of 0.96 was reached using the perfect learning and the RVFL method, for example for the two-dimensional data whitening and parameter $J = 50$, $\lambda = 0.2$. The imperfect learning leads to the critical sensitivity

up to 0.98, for example for the three-dimensional data whitening, $J = 50$, and $\lambda = 0.2$. Even the high critical sensitivity in the case of the perfect learning was increased by the imperfect learning. The results for $J = 50$ and the different settings of parameter α are captured in Figure 6.6.

The MM method and perfect learning was not able to maximize the critical sensitivity above zero due to the character of the given dataset. Three-dimensional data whitening led to critical sensitivity of 0.95. Therefore, in this case, imperfect learning significantly improved the classifier results.

Using the traditional Parzen method, we reached the critical sensitivity at 0.96 with the RAO method preprocessing and parameter $h = 0.66$. When we use the imperfect learning, the highest critical sensitivity is 0.96, for example for the RAO method preprocessing and parameter $h = 0.53$. In this case, the highest critical sensitivity is the same for the perfect and the imperfect learning. All the results for the different settings of parameter h are captured in Figure 6.7.

The perfect learning using the LQ method led to the critical sensitivity of 0.96 reached for example for the RAO method preprocessing and the parameter $k = 5$. In the case of the imperfect learning the highest critical sensitivity is 0.94 for the RAO method preprocessing and the parameter $k = 1$. The LQ method provided better results for the perfect cases and our imperfect learning was not able to reach higher critical sensitivity. The critical sensitivities for the different settings of parameter k are captured in Figure 6.8.

The highest critical sensitivity by using the K-means classifier is 0.96, for example for the RAO method preprocessing and $k = 14$. When we use imperfect learning, the critical sensitivity is up to 0.96, for example for the RAO method preprocessing and $k = 1$. This classifier method provided the same results of the critical sensitivity for the perfect and the imperfect cases. The obtained results for the different settings of parameter k are captured in Figure 6.9.

To present comparable results for the SOM we focus on different sizes of hexagonal maps (H). The size of the map is presented as a total number of nodes in the map. Firstly, we focus on the traditional Kohonen learning and we use a novel diffusion strategy of learning.

In the case of the traditional Kohonen learning, we reached the critical sensitivity of 0.92. This value was reached by the RAO method preprocessing, smaller sizes of maps, and using the three epoch learning strategy. The imperfect learning led to the critical sensitivity of 0.94, which was reached for the map with nineteen nodes. In this case, the use of the imperfect learning improved the classifier results. The final values of sensitivities for the different sizes of maps are captured in Figure 6.10.

In the case of the diffusion learning, we reached the critical sensitivity of 0.94. This value was reached using the RAO method preprocessing, smaller sizes of maps, and a single epoch learning strategy. The imperfect learning led to the critical sensitivity of 0.94, which was reached for the map with seven nodes. The perfect and the imperfect learning reached the same critical sensitivity. The re-

sults for the different sizes of maps are captured in Figure 6.11.

The SLINK method was implemented as a special case of the DBSCAN, where the clusters are only pairs. The highest critical sensitivity in the case of the perfect learning was reached at 0.88 using the RAO method preprocessing and for example $\epsilon = 0.15$. The imperfect learning was able to reach the critical sensitivity up to 0.96, for example for one-dimensional data whitening and $\epsilon = 0.07$. The results for the different values of ϵ are captured in Figure 6.12.

In the case of the DBSCAN and the perfect learning, the highest critical sensitivity reached 0.86 for example for the RAO method preprocessing, minimal five points in the cluster, and $\epsilon = 0.17$. For the same minimal size of the cluster, the RAO method preprocessing, and $\epsilon = 0.2$ we reached the critical sensitivity at 0.92. In this case, the imperfect learning improved the classifier results significantly. The final values of the sensitivities for different values of ϵ are captured in Figure 6.13. The results for the bigger clusters, up to seven points in cluster, are depicted in Figures 6.14 and 6.15.

The previous results generally show that higher critical sensitivities are reached in the case of the imperfect learning strategies. The highest improvement is seen for the RR, the KRR, the LDA, the QDA, the RVFL, the MM, the SLINK, and the DBSCAN. There were similar results for the Parzen method, the K-means, and the SOM. In the case of the LQ, the perfect learning strategy provided higher critical sensitivity.

6.3 Other Testing Datasets Results

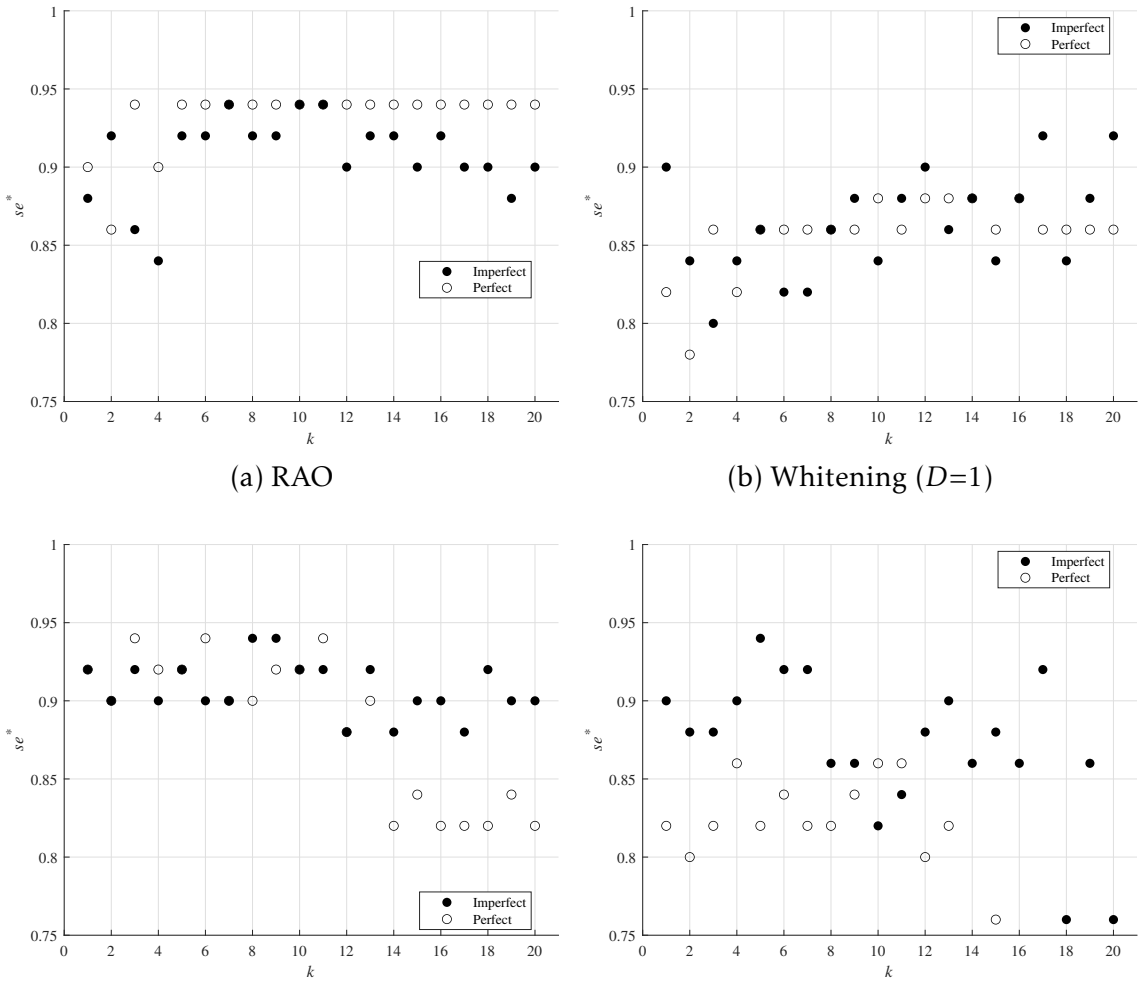
To demonstrate the results of the proposed approach we have selected nine additional classification tasks [73] next to the basic iris dataset presented previously. The list of benchmark datasets is included in Table 6.2 together with the number of patterns, their dimensionality, and the number of classes.

The main aim is to present whether the proposed imperfect classifier is comparable to the perfect one. The methods used for the comparison are selected based on the iris results. We have selected the methods that provided high values of the critical sensitivity, i.e. the KRR, the LDA, the QDA, the RVFL, the MM, the Parzen method, the K-means, and the SLINK. These are the methods providing the critical sensitivities higher than 0.95 for the iris dataset and the imperfect learning. The complete results of this subset of methods are summarized in Table 6.3. The individual results are commented on as follows.

There are the same or better results for all methods for the imperfect learning and the Cancer (Wisconsin) dataset [74]. The best results for the imperfect learning were reached for the QDA, the Parzen method, the K-means, and the SLINK, when the critical sensitivity reached 0.98 and it was higher than in the case of the perfect learning in all the cases.

Table 6.1: IRIS Flower Classification

Method	se^*	Perfect Learning Parameters	se^*	Imperfect Learning Parameters
k-NN	0.94	RAO; k=3, 5-20 D=2; k=3,6,11	0.94	RAO, k=7,10,11 D=3, k=5
RR	0.68	RAO, $\lambda = 0$	0.82	RAO, $\lambda = 0$ D=3, $\lambda = 0$
KRR	0.94	RAO, $\sigma \in [0.5, 1.0]$ D=2, $\sigma \in [0.3, 0.4]$ D=3, $\sigma \in [0.5, 1.0]$	0.96	$D = 2$, $\sigma \in [0.1, 0.8]$ D=3, $\sigma \in [0.6, 0.7]$
LDA	0.96	RAO, $\lambda \in [0.01, 0.13]$ D=3, $\lambda \in [0, 0.04]$	0.98	D=2, $\lambda \in [0, 0.02]$ D=3, $\lambda \in [0, 0.01]$
QDA	0.96	D=3, $\lambda \in [0, 0.02]$	0.98	D=3, $\lambda \in [0, 0.02]$
RVFL	0.96	RAO, $J = 50$, $\lambda \in [0, 0.05]$ D=2, $J = 50$, $\lambda \in [0.1, 0.4]$ D=3, $J = 50$, $\lambda \in [0.1, 0.4]$	0.98	D=3, $J = 50$, $\lambda \in [0.2, 0.4]$
MM	0.00	-	0.95	D=3
Parzen	0.96	RAO, $h \in [0.6, 0.7]$	0.96	RAO, $h \in [0.5, 0.8]$ D=2, $h \in [0.55, 0.65]$
LQ	0.96	RAO, k=5	0.94	RAO, k=1,2,4,5,6,8,12,14 D=2, k=1,9
K-means	0.96	RAO, k=14,16,23,26	0.96	D=2, k=22
SOM (Kohonen)	0.92	RAO	0.92	RAO
SOM (Diffusion)	0.94	RAO	0.94	RAO, D=1
SLINK	0.88	RAO, $\epsilon \in [0.1, 0.2]$	0.96	D=1, $\epsilon \in [0.05, 0.1]$
DBSCAN	0.86	RAO, $minpts = 5$, $\epsilon \in [0.15, 0.2]$	0.92	RAO, $minpts = 5$, $\epsilon \in [0.1, 0.3]$



(a) RAO

(b) Whitening (D=1)

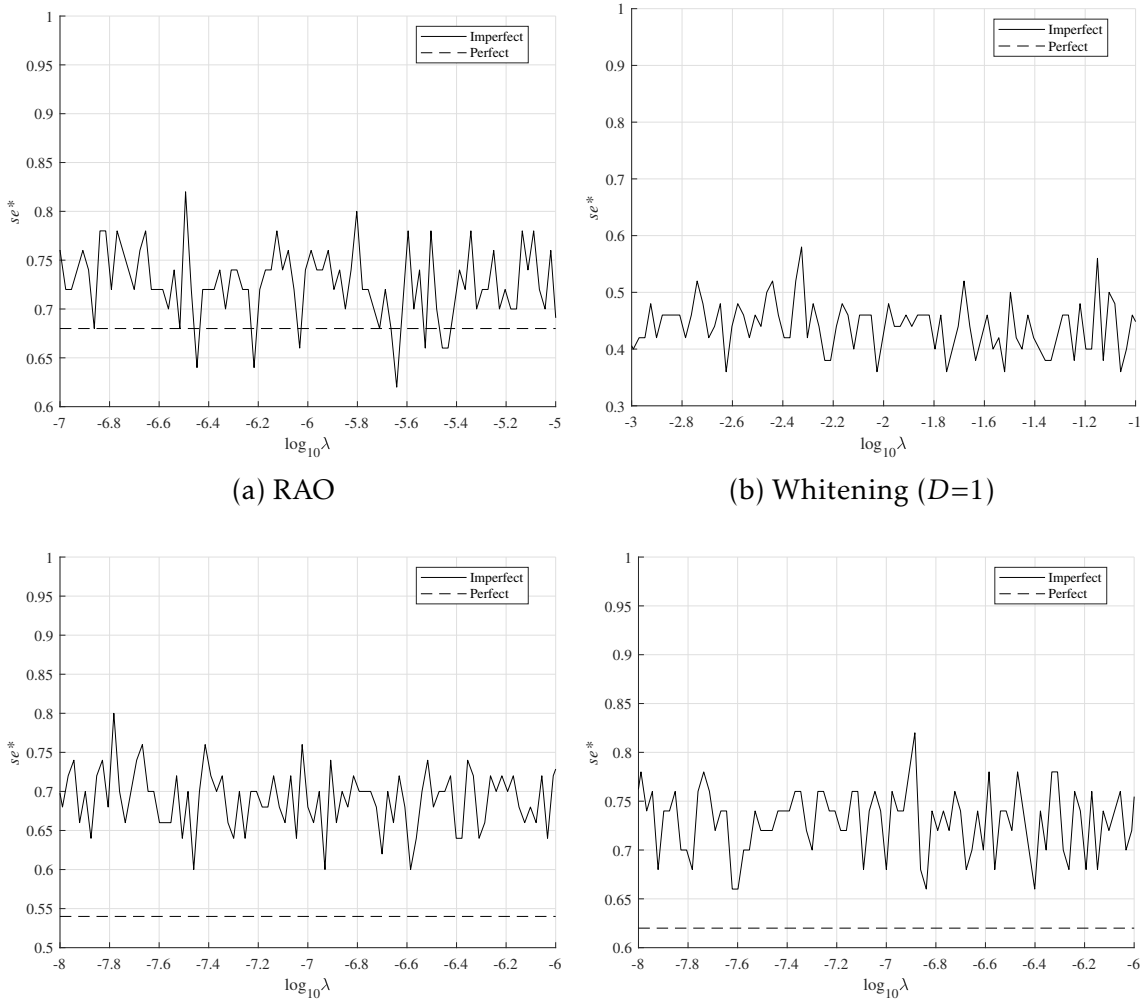
(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.1: Critical sensitivity of k-NN for IRIS Flower

Table 6.2: Datasets for Assessment

Dataset	m	n	N	m^*	n^*
Iris [73]	150	4	3	100	3
Cancer (Wisconsin) [74]	699	9	2	466	6
Digits [73]	3823	64	10	2550	48
Cryotherapy [75, 76]	90	6	2	60	4
Wine [73]	178	13	3	120	10
Glass [73]	214	9	2	143	6
Ionosphere [73]	351	33	2	234	22
Cancer (Coimbra) [77]	116	10	2	78	7
Transfusion [78]	748	4	2	499	3
Liver [73]	345	6	2	230	4



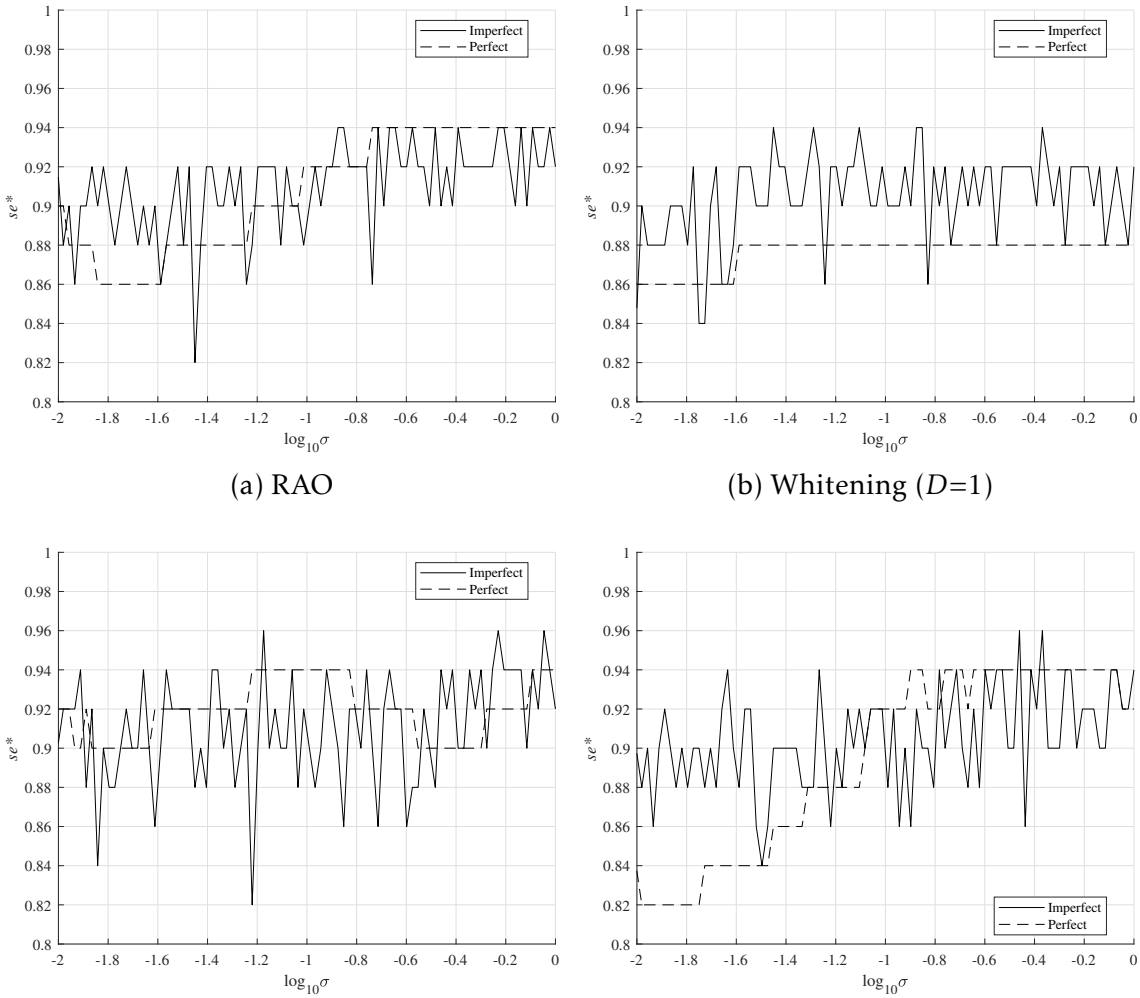
(a) RAO

(b) Whitening (D=1)

(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.2: Critical sensitivity of RR for IRIS Flower



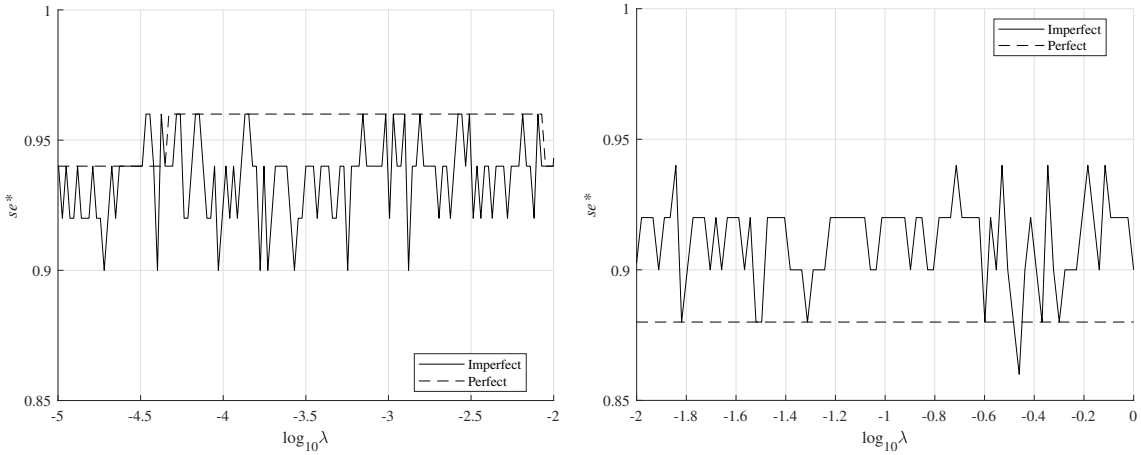
(a) RAO

(b) Whitening (D=1)

(c) Whitening (D=2)

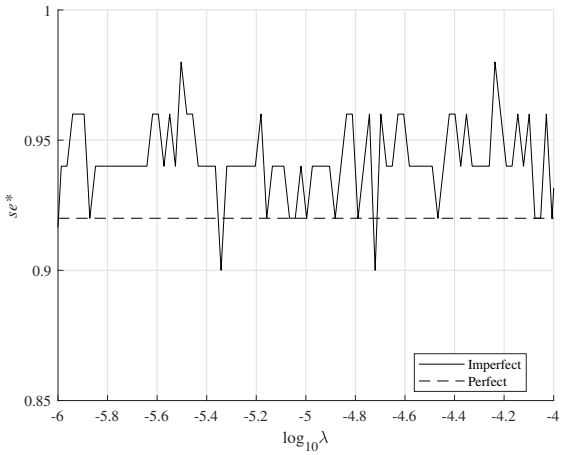
(d) Whitening (D=3)

Figure 6.3: Critical sensitivity of KRR for IRIS Flower

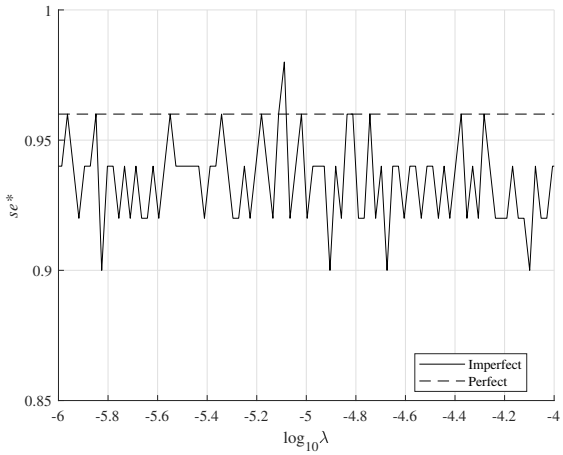


(a) RAO

(b) Whitening (D=1)

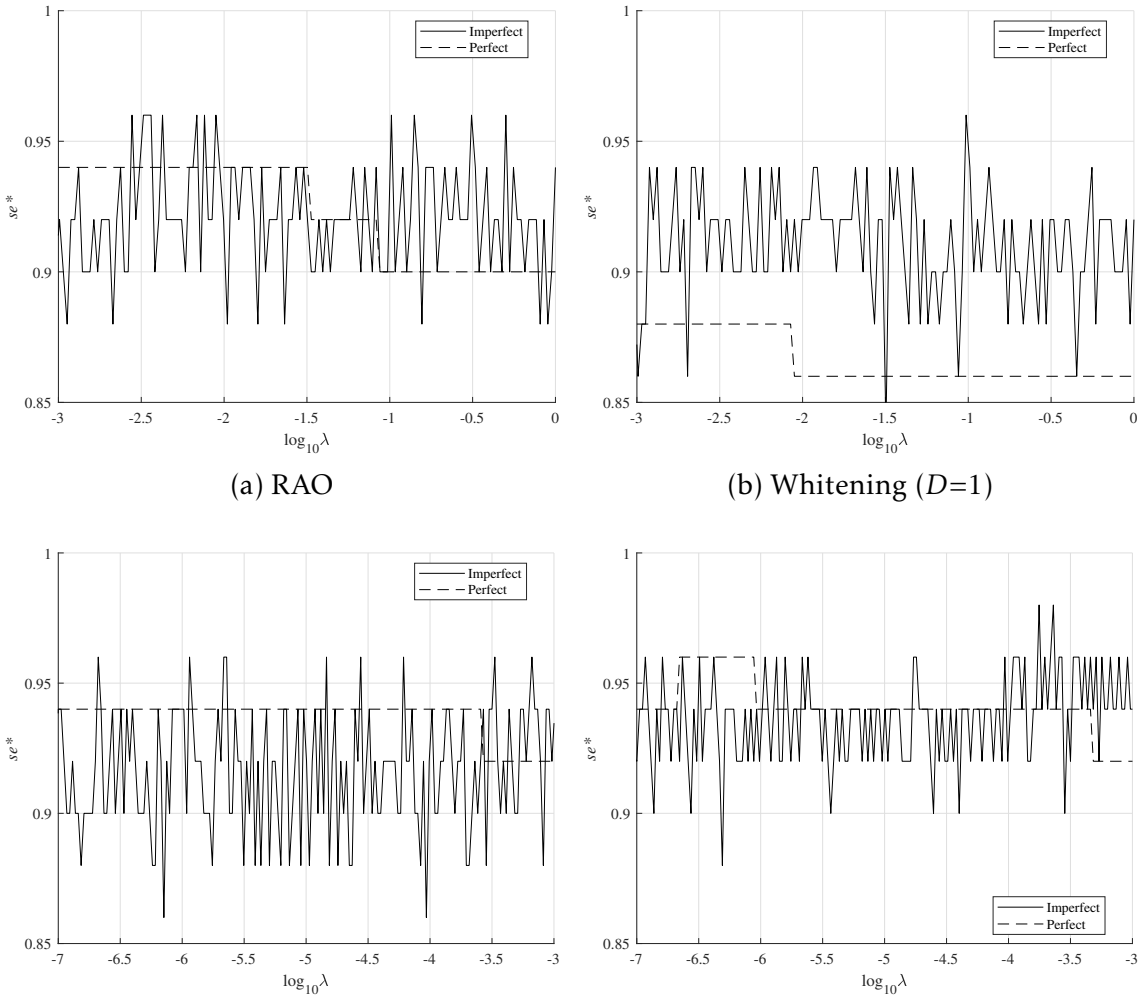


(c) Whitening (D=2)



(d) Whitening (D=3)

Figure 6.4: Critical sensitivity of LDA for IRIS Flower



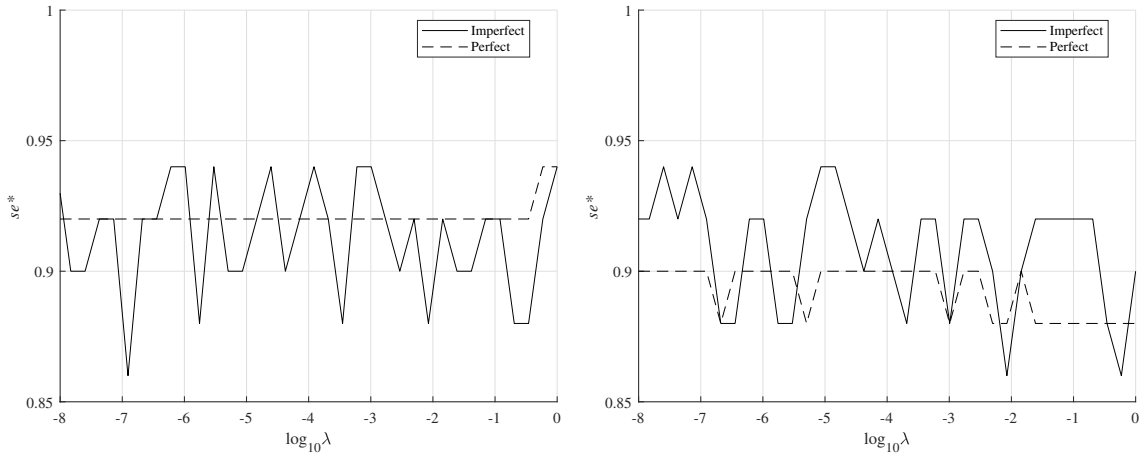
(a) RAO

(b) Whitening (D=1)

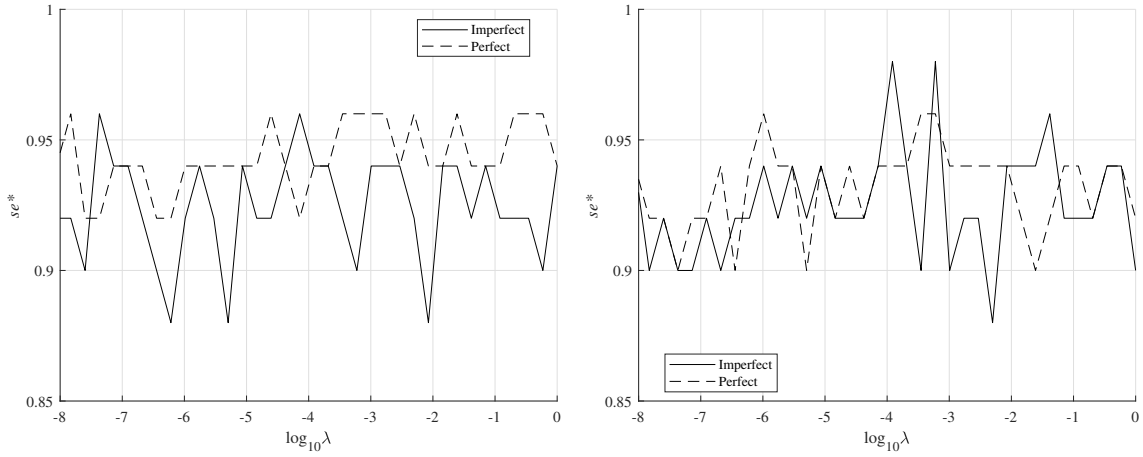
(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.5: Critical sensitivity of QDA for IRIS Flower

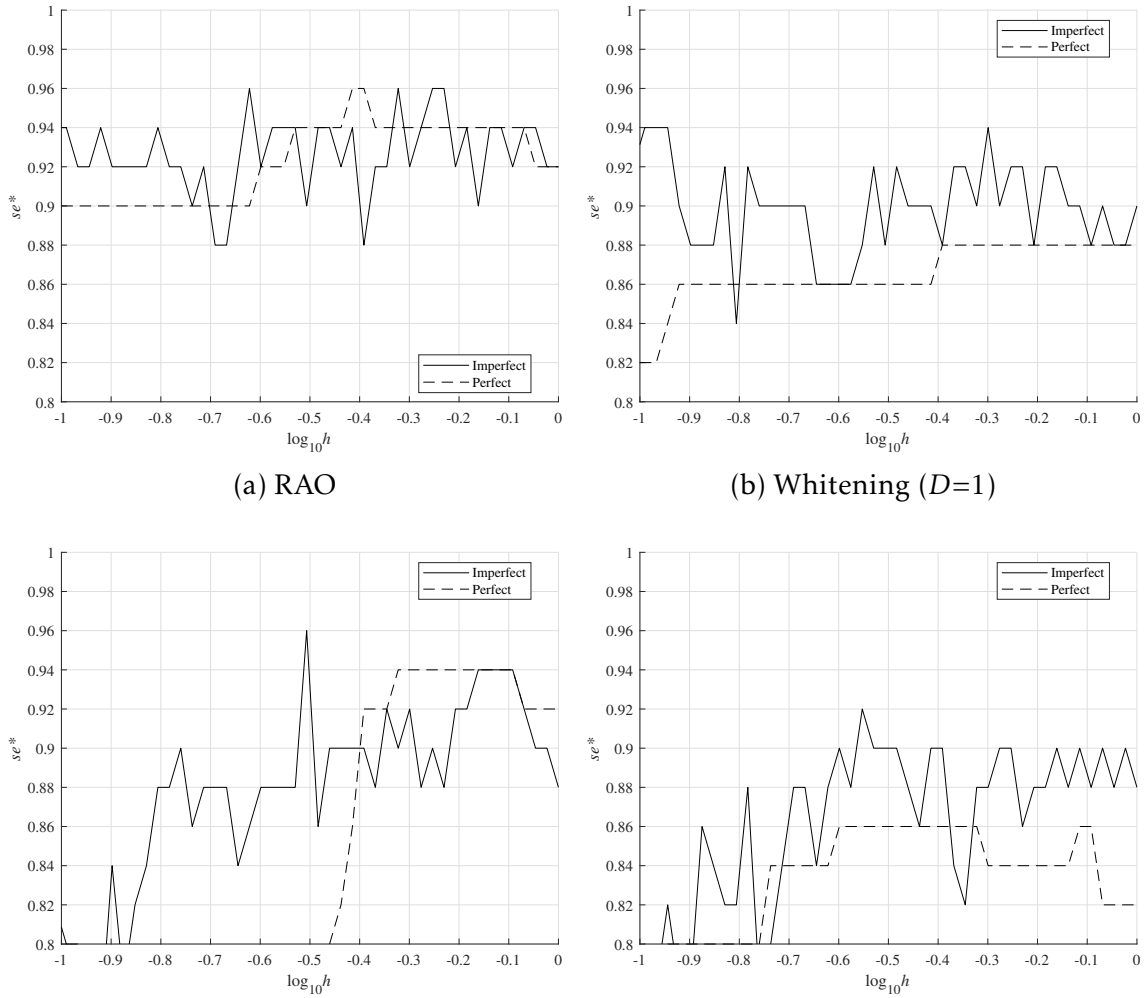


(a) RAO (b) Whitening (D=1)



(c) Whitening (D=2) (d) Whitening (D=3)

Figure 6.6: Critical sensitivity of RVFL (J=50) for IRIS Flower



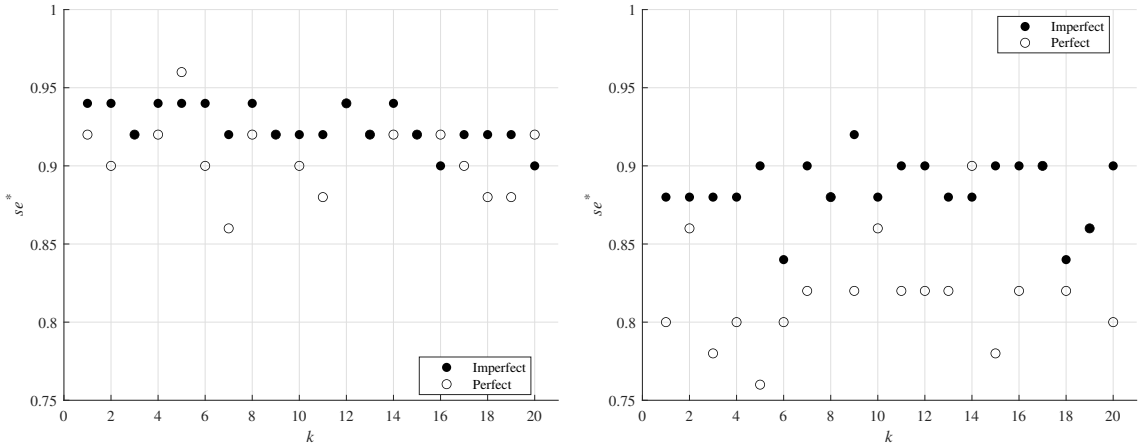
(a) RAO

(b) Whitening (D=1)

(c) Whitening (D=2)

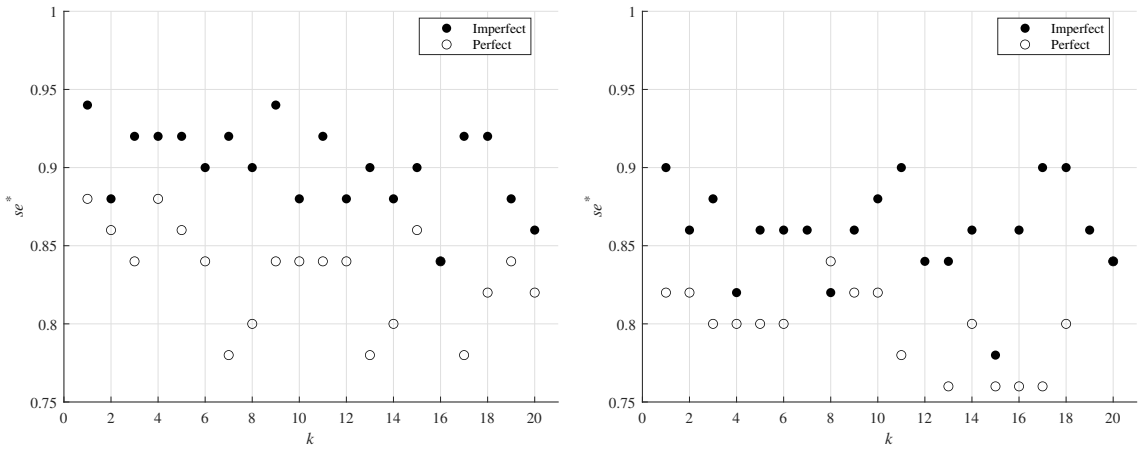
(d) Whitening (D=3)

Figure 6.7: Critical sensitivity of Parzen method for IRIS Flower



(a) RAO

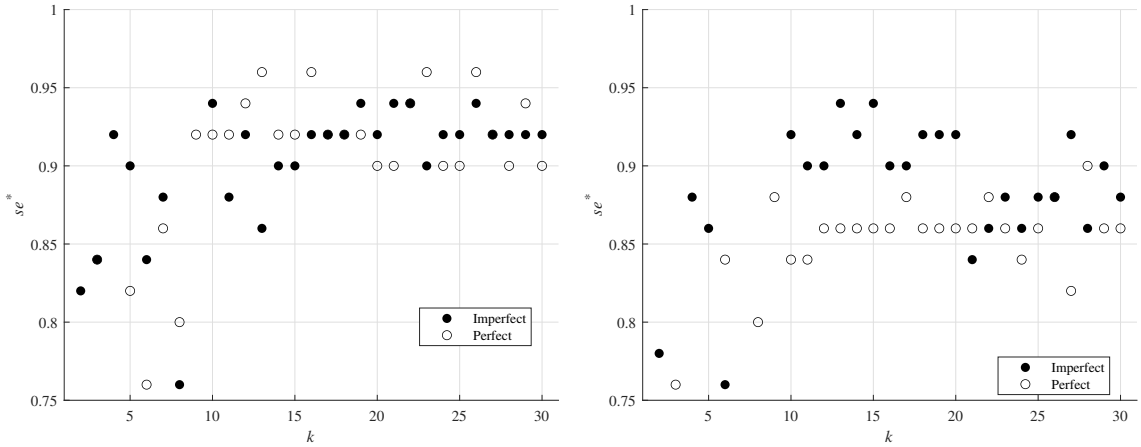
(b) Whitening (D=1)



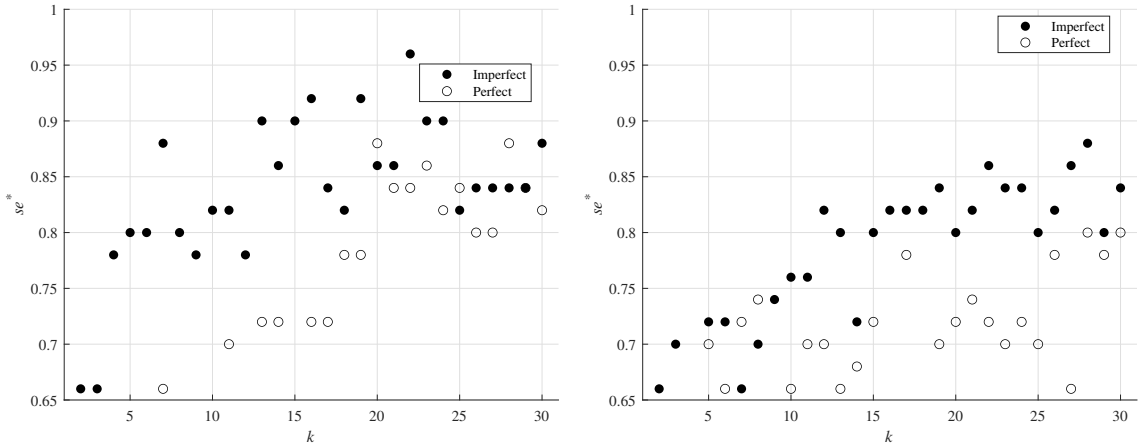
(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.8: Critical sensitivity of LQ for IRIS Flower

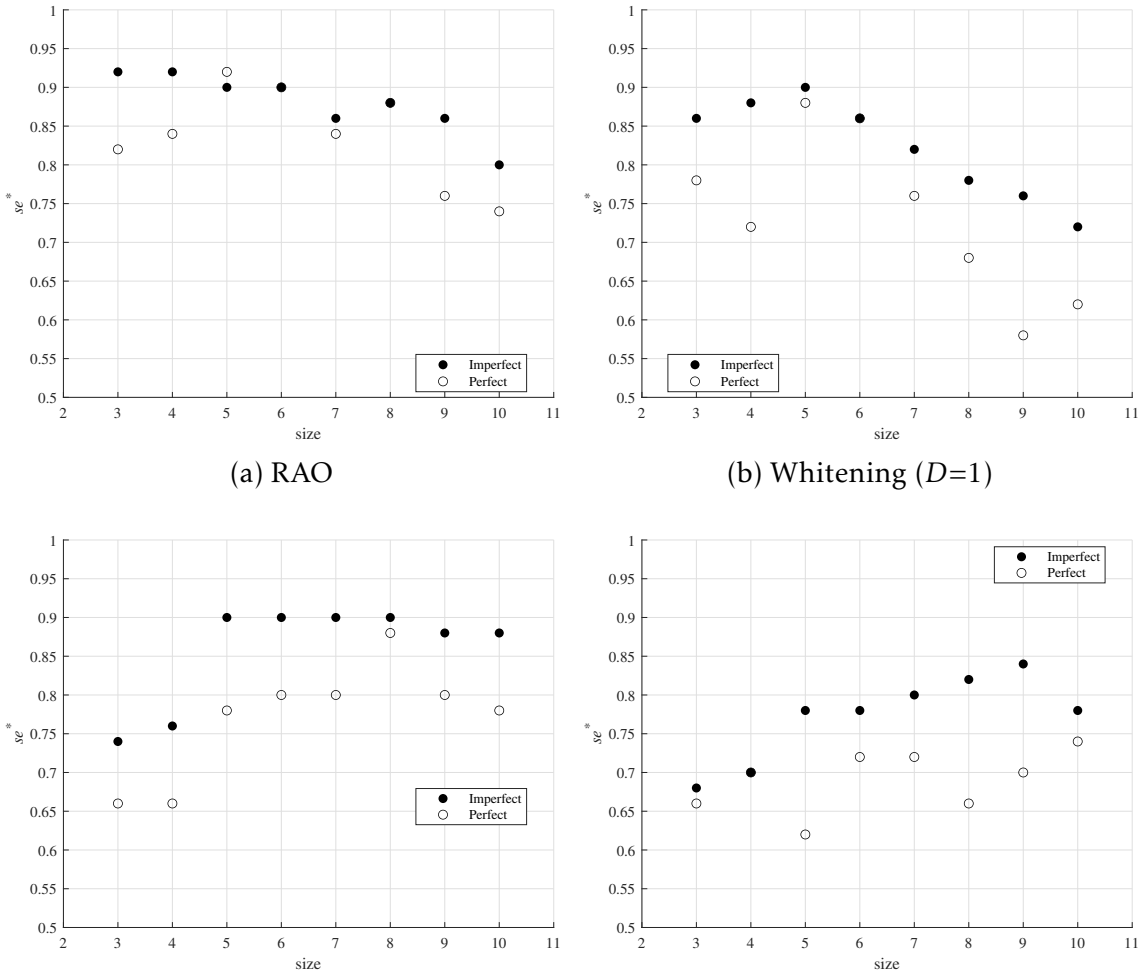


(a) RAO (b) Whitening (D=1)



(c) Whitening (D=2) (d) Whitening (D=3)

Figure 6.9: Critical sensitivity of K-means for IRIS Flower



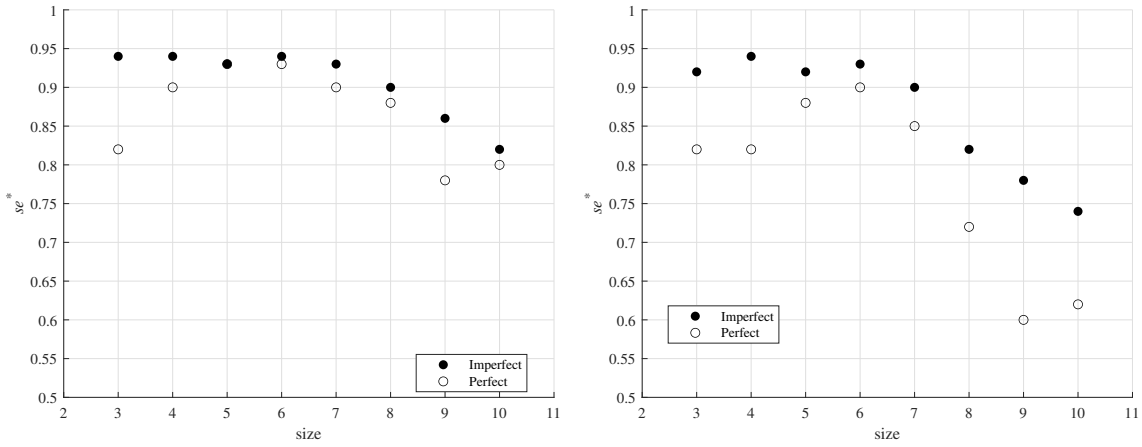
(a) RAO

(b) Whitening (D=1)

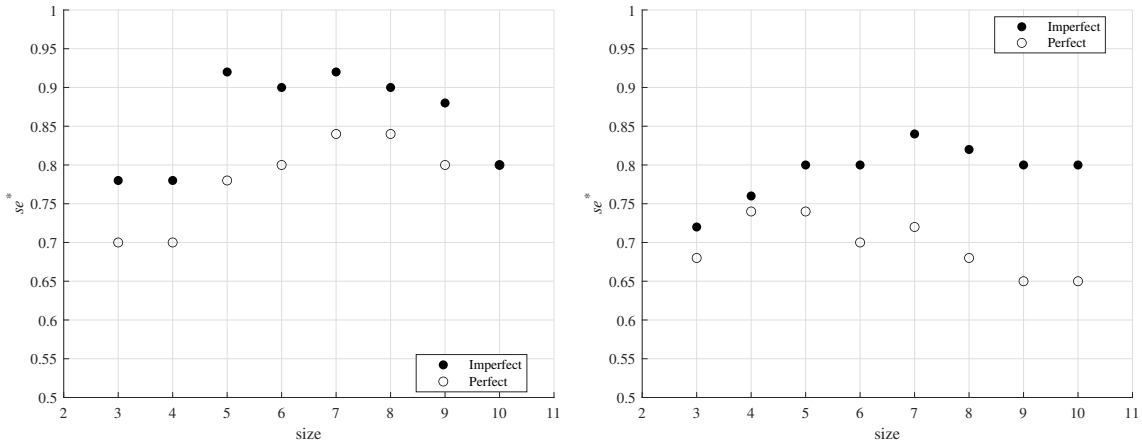
(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.10: Critical sensitivity of Kohonen SOM with Hexagonal Topology for IRIS Flower

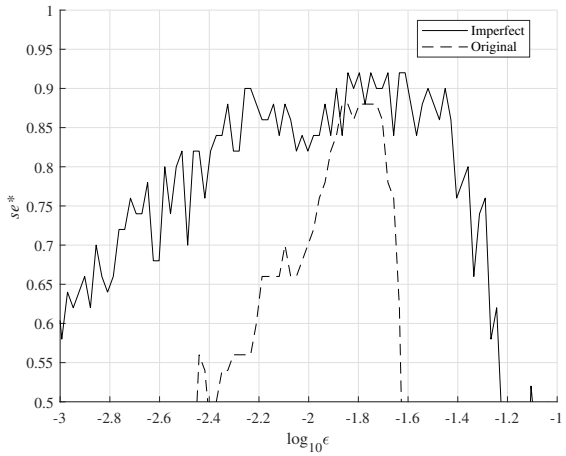


(a) RAO (b) Whitening (D=1)

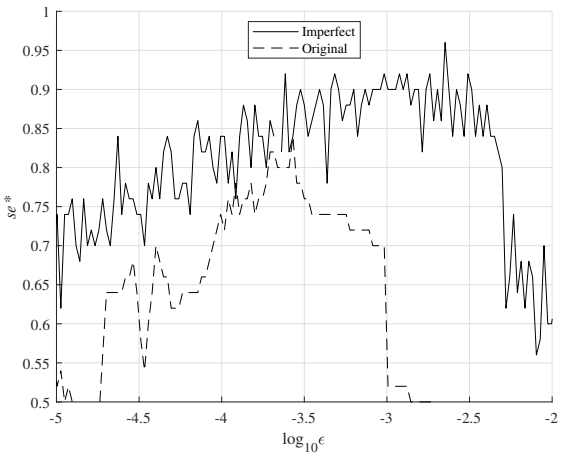


(c) Whitening (D=2) (d) Whitening (D=3)

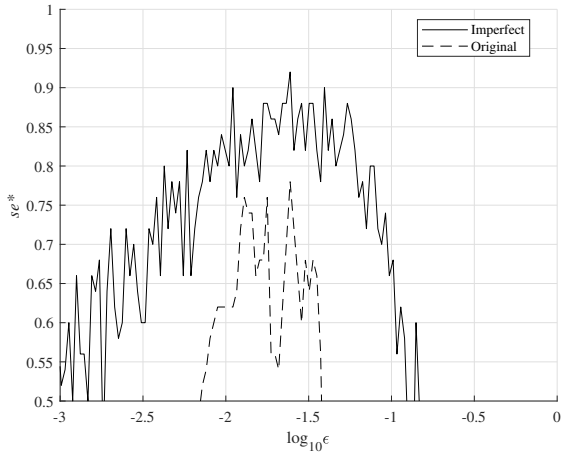
Figure 6.11: Critical sensitivity of Diffusion SOM with Hexagonal Topology for IRIS Flower



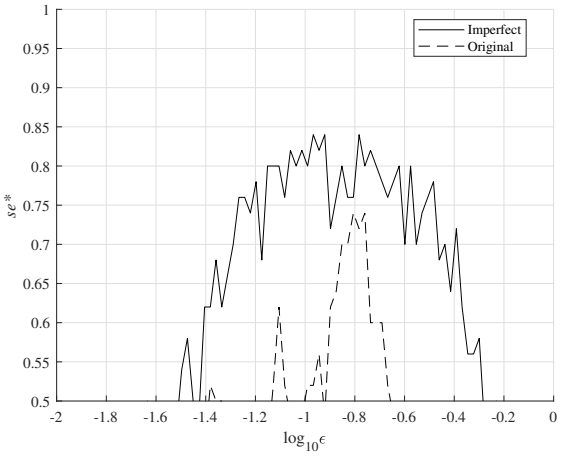
(a) RAO



(b) Whitening (D=1)

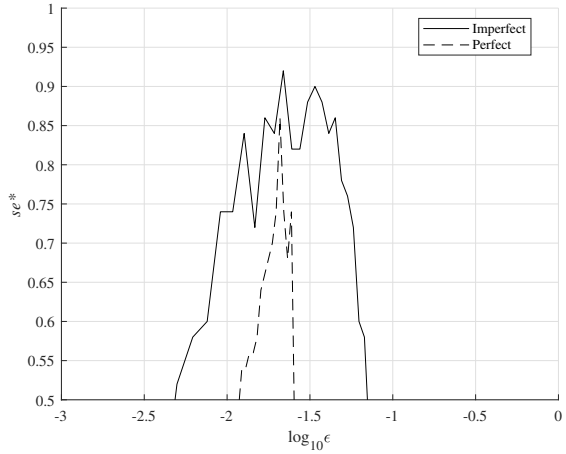


(c) Whitening (D=2)

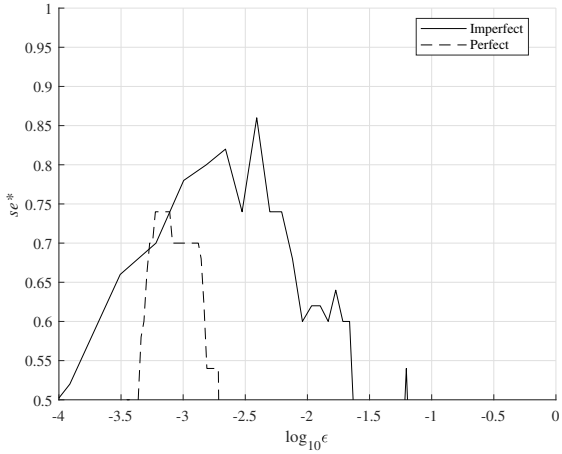


(d) Whitening (D=3)

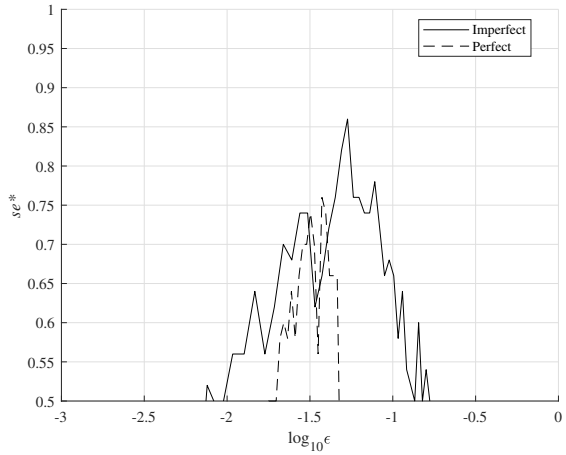
Figure 6.12: Critical sensitivity of SLINK for IRIS Flower



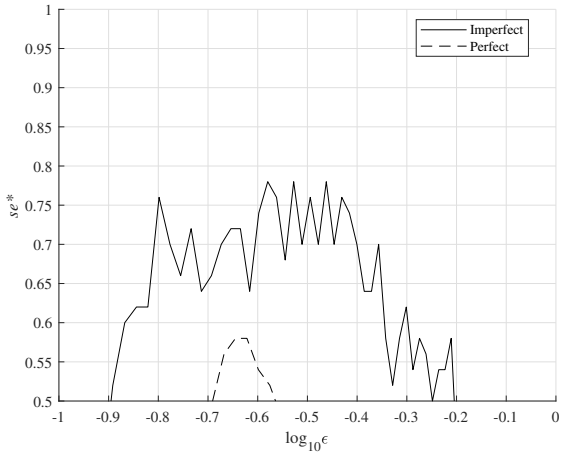
(a) RAO



(b) Whitening (D=1)

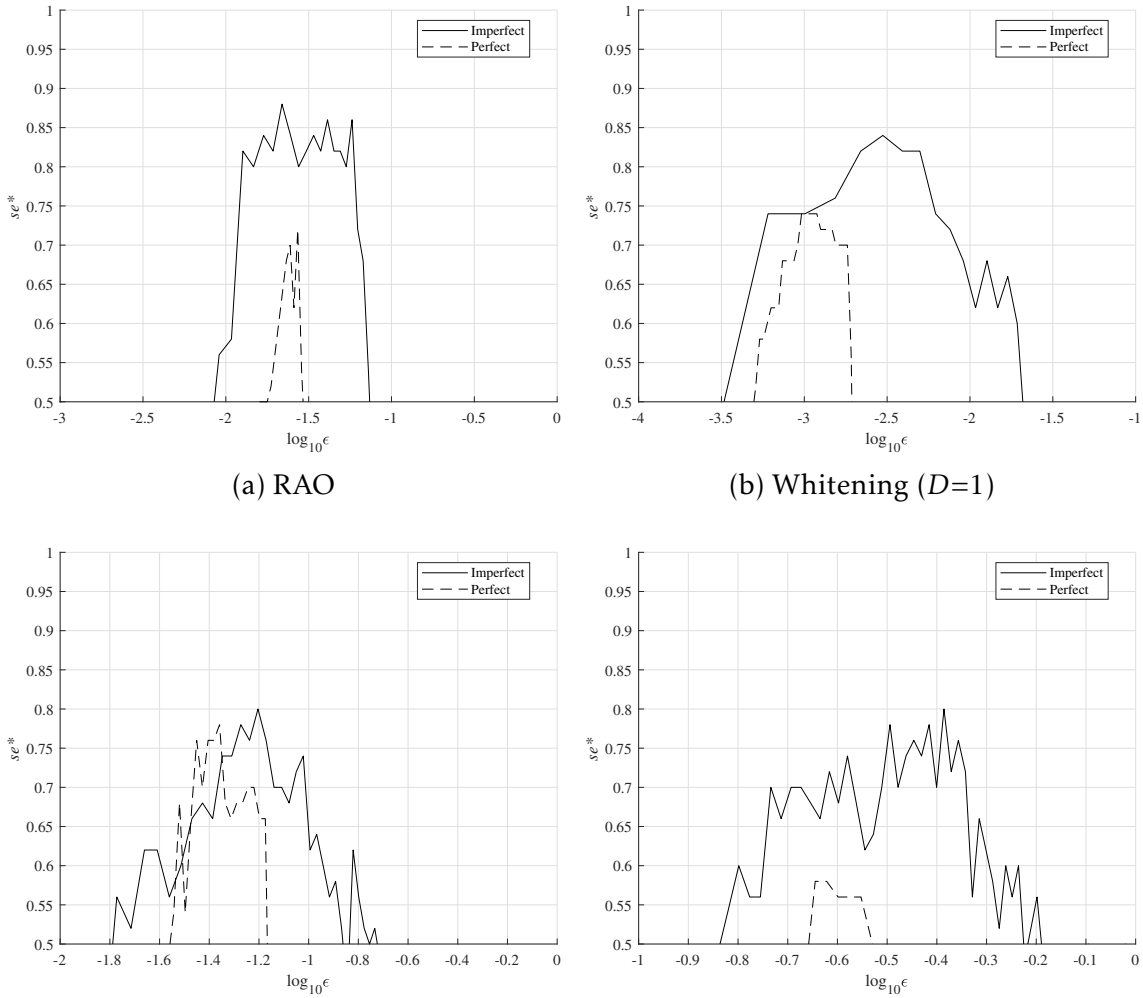


(c) Whitening (D=2)



(d) Whitening (D=3)

Figure 6.13: Critical sensitivity of DBSCAN (D+1) for IRIS Flower



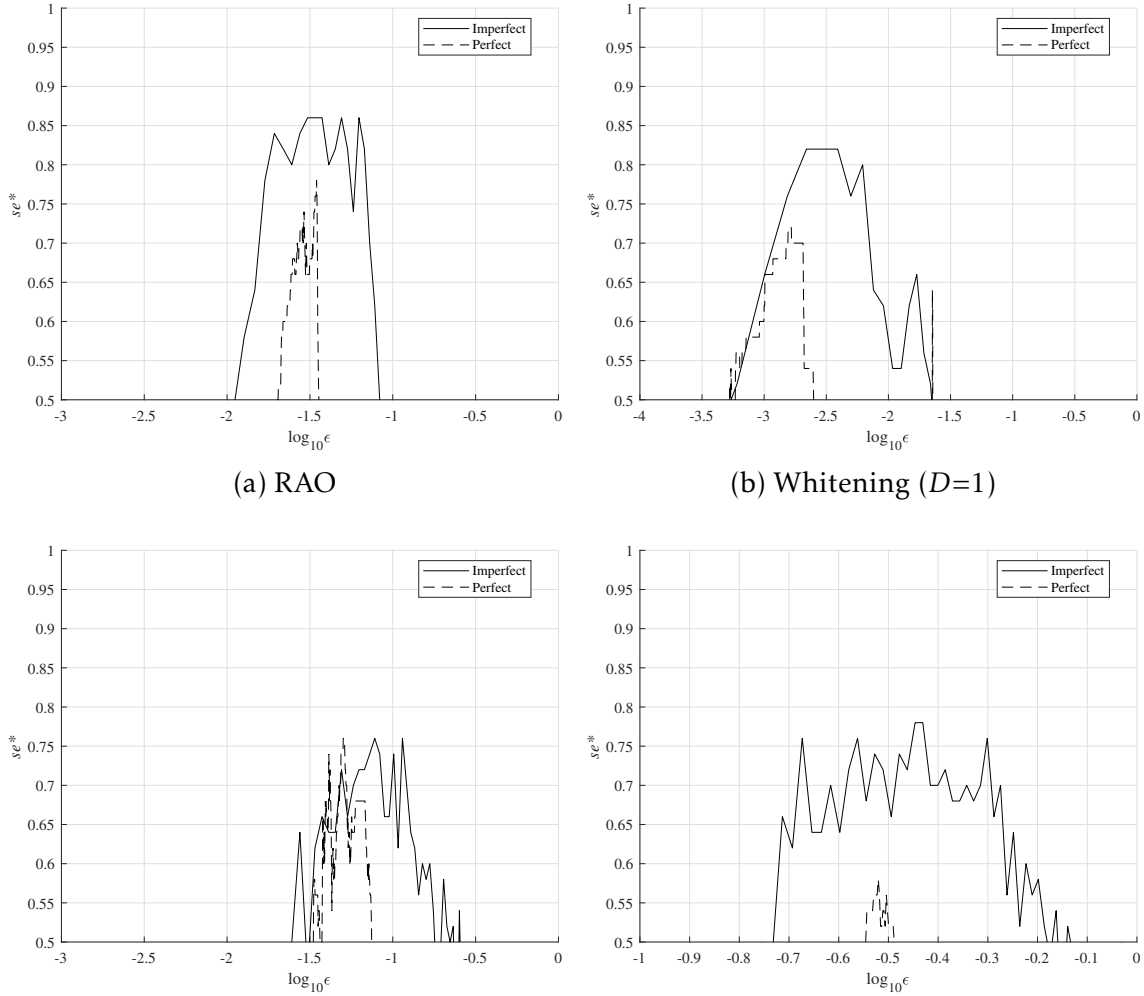
(a) RAO

(b) Whitening (D=1)

(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.14: Critical sensitivity of DBSCAN (D+2) for IRIS Flower



(a) RAO

(b) Whitening (D=1)

(c) Whitening (D=2)

(d) Whitening (D=3)

Figure 6.15: Critical sensitivity of DBSCAN (D+3) for IRIS Flower

Table 6.3: Critical Sensitivity of Imperfect Learning
(Perfect Learning Results in Brackets)

Dataset	KRR	LDA	QDA	RVFL	MM	Parzen	K-means	SLINK
Iris	0.96 (0.94)	0.98 (0.96)	0.98 (0.96)	0.98 (0.96)	0.95 (0.00)	0.96 (0.96)	0.96 (0.96)	0.96 (0.92)
Cancer (Wisconsin)	0.97 (0.97)	0.97 (0.94)	0.98 (0.97)	0.97 (0.97)	0.97 (0.94)	0.98 (0.97)	0.98 (0.97)	0.98 (0.96)
Digits	0.85 (0.79)	0.96 (0.89)	0.96 (0.93)	0.94 (0.90)	0.92 (0.88)	0.97 (0.96)	0.60 (0.46)	0.80 (0.72)
Cryotherapy	0.98 (0.94)	0.92 (0.71)	0.88 (0.50)	0.95 (0.81)	0.93 (0.71)	0.98 (0.88)	0.94 (0.83)	0.98 (0.95)
Wine	0.93 (0.77)	0.93 (0.75)	0.94 (0.77)	0.92 (0.81)	0.40 (0.31)	0.75 (0.73)	0.89 (0.73)	0.64 (0.52)
Glass	0.96 (0.92)	0.94 (0.78)	0.90 (0.90)	0.94 (0.94)	0.92 (0.88)	0.95 (0.94)	0.96 (0.95)	0.94 (0.92)
Ionosphere	0.79 (0.78)	0.79 (0.75)	0.84 (0.79)	0.84 (0.82)	0.72 (0.72)	0.83 (0.77)	0.77 (0.75)	0.80 (0.79)
Transfusion	0.43 (0.38)	0.69 (0.63)	0.69 (0.65)	0.41 (0.39)	0.73 (0.64)	0.62 (0.62)	0.65 (0.66)	0.62 (0.62)
Cancer (Coimbra)	0.71 (0.58)	0.72 (0.47)	0.75 (0.34)	0.75 (0.58)	0.67 (0.48)	0.73 (0.61)	0.65 (0.60)	0.71 (0.61)
Liver	0.62 (0.55)	0.66 (0.57)	0.63 (0.27)	0.66 (0.62)	0.55 (0.17)	0.65 (0.67)	0.63 (0.63)	0.60 (0.62)
Leaves	0.97 (0.94)	0.99 (0.88)	0.97 (0.89)	0.95 (0.92)	0.94 (0.70)	0.78 (0.79)	0.82 (0.63)	0.73 (0.62)
Crisis	0.90 (0.83)	0.92 (0.85)	0.94 (0.83)	0.90 (0.82)	0.86 (0.81)	0.93 (0.89)	0.94 (0.83)	0.88 (0.67)

As a dataset classifying into ten classes we used the Digits dataset [73]. This robust dataset shows increases in the critical sensitivities for all the imperfect learning methods. The highest critical sensitivity is 0.97 and it was reached by using the imperfect learning strategy and Parzen method.

For the Cryotherapy dataset [75] the imperfect learning was able to reach the critical sensitivity at 0.98 for the KRR, the Parzen method, and the SLINK. In all the cases we see better results compared to the perfect learning.

The critical sensitivity higher than 0.90 for the Wine dataset [73] was reached for the KRR, the LDA, and the QDA methods. The perfect learning strategies provided a lower level of the critical sensitivity in all the cases.

Using the imperfect learning strategy and Glass dataset [73], we were able to reach the critical sensitivity at 0.96 for the KRR and the K-means. The perfect learning strategy led to the same critical sensitivity as the imperfect one when using the QDA and the RVFL methods.

The highest critical sensitivity for the Ionosphere dataset [73] was 0.84 in the case of the QDA and the RVFL methods when the imperfect learning strategy was applied. In this case, the imperfect learning strategy led to at least the same results as the perfect learning.

The imperfect learning was able to reach the critical sensitivity at only 0.73 for the Transfusion dataset using the MM. In the case of the K-means method better results were reached for the perfect learning strategy.

Cancer (Coimbra) dataset [77] reached the critical sensitivity at 0.75 for the QDA and the RVFL when using an imperfect learning strategy. The sensitivities were higher for all the methods in comparison to the perfect learning.

The lowest levels of critical sensitivities were reached for the Liver dataset [73]. The highest critical sensitivity was only 0.66 for the LDA and the RVFL when using the imperfect learning strategy. In the case of the Parzen method and the SLINK, the perfect learning provided higher values of the critical sensitivity compared to the imperfect learning.

There is no clear suggestion for an individual universal method for all the presented datasets. Based on the first six datasets presented in Table 6.3, where the critical sensitivities higher than 0.90 were reached, we can summarize the results as follows. The critical sensitivity above 0.90 was reached for the LDA and the RVFL in all cases. In the case of all datasets assessment, we would be not able to provide any method with the critical sensitivity higher than 0.90.

For first six datasets presented in Table 6.3, the sensitivities higher than 0.85 were reached for the KRR, the LDA, the QDA, and the RVFL methods.

Therefore, we can generally propose the imperfect learning as a method which is able to reach higher critical sensitivity in most cases. In the case of method, we propose the LDA and the RVFL.

6.4 Real Application

In this section, we present the results for two real world datasets for classification, which are not so famous and popular compared to the previous ones presented in Table 6.2. As a practical example of how the new classifier works, we decided to present the results for following classification tasks. The first is the dataset representing tree leaves [79], the second is based on the economic indicators used for pre and post crisis assessment [70]. The first dataset is focused on the recognition of ten tree leaves, the second task classifies the economic indicators into two groups, before and after crisis.

6.4.1 Leaves

The first dataset is generated from the photographs of the leaves, it is so called Czech leave dataset and it was presented in [79, 80]. This dataset represents the description of the real leaves of ten trees: Ginkgo Biloba, Acer Platanoides, Betula Pendula, Salix Fragilis, Fagus Sylvatica, Populus Canadensis, Tilia x Vulgaris, Quercus Robur, Populus Tremula, and Prunus Avium. The total number of features is two hundred and eighty six, formed by the functional values of eleven harmonic components sampled for twenty six frequencies. The total number of patterns is nine hundred. Therefore, it means $m = 900, n = 286$, and $N = 10$.

The pattern set was formed by taking photographs from nine views – one from the top and eight from various angles and distances. There are differences between harmonics of the same leaf photographed at various angles and distances. Affine invariant characteristics were formed from this primary representation. The advantage of these characteristics is in their preservation during translation, rotation, scaling and other geometric affine transformation. The detail description regarding the feature extraction and its characteristics is available in [80].

For comparison of the perfect and the imperfect learning we follow our approach and we apply the basic dimensionality reduction, the RAO method preprocessing and the data whitening. In case of the imperfect learning, we deal with the reduction of number of patterns to six hundred, and the reduction of number of features to one hundred and ninety one. Therefore, it means $m^* = 600$ and $n^* = 191$. The number of repeated learning was sufficient at level of fifty. The classifier results for seven selected methods providing the best results are presented in the bottom of the Table 6.3.

The highest value of critical sensitivity reached 0.99 in the case of the LDA method and the proposed imperfect learning strategy. Therefore, the suggestion proposed in Section 6.3 works for this task. Moreover, the results show the imperfect learning over-performing the perfect strategy for all the presented methods.

6.4.2 Crisis

The crisis dataset is based on economic indicators and was presented in [70]. This dataset represents behaviour of nine indicators of twenty eight European countries in given years. The indicators are total population, unemployment rate, gross domestic product, private final consumption, gross fixed capital formation, domestic demand, exports of goods and services, imports of goods and services, and gross national savings. The total number of features is nine, total number of patterns is six hundred and forty four. For each country we have included indicators in twenty three years, from which sixteen classified as before and seven as after crisis. Therefore, it means $m = 644, n = 9$, and $N = 2$. The pattern set was formed by taking indicators from the annual report as proposed in [70].

For comparison of the perfect and the imperfect learning we follow our approach and we apply the basic dimensionality reduction, the RAO method preprocessing method and the data whitening. In case of the imperfect learning, we deal with the reduction of number of patterns to four hundred and thirty, and the reduction of number of features to six. Therefore, it means $m^* = 430$ and $n^* = 6$. The number of repeated learning was sufficient at level of fifty. The classifier results for seven selected methods providing the best results are presented in the bottom of the Table 6.3.

The highest value of the critical sensitivity reached 0.94 in the case of the QDA and the K-means method for the proposed imperfect learning strategy. The results show the imperfect learning over-performing the perfect strategy for all presented methods. The methods proposed in Section 6.3 succeed in this case as well. Moreover, the imperfect learning reached critical sensitivity above 0.90 for the most methods.

Chapter 7

Implementation in Matlab

We used Matlab for the proposed classifier implementation. We have proposed general implementation of selected methods via separate functions for learning, classification, and validation. In this chapter we present brief summary of used functions together with basic description.

Table 7.1: Supporting and Learning Functions

Name	Description	Parameters
`DWH`	Data whitening (1.4.1)	$D_{\text{red}} \in \mathbb{N}$
`RAO`	RAO method (1.4.2)	-
`DBSCANLEARN`	DBSCAN (1.3.1)	$\epsilon > 0, minpts > n$
`DBSCANLEARN`	SLINK (1.3.1)	$\epsilon > 0, minpts = 2$
`DIFFSOMLEARN`	Diffusion SOM (1.3.2)	H, a, b
`KMEANSLEARN`	K-means (1.3.1)	$\mathrm{K} \geq 2$
`KNNLEARN`	k-NN (1.2.1)	$k \in \mathbb{N}$
`KOHONENLEARN`	SOM (1.3.2)	H, k, R
`KRRLEARN`	KRR (1.2.2)	$\lambda > 0, \sigma > 0$
`LDALEARN`	LDA (1.2.1)	$\lambda > 0$
`LQLEARN`	LQ (1.2.1)	$k \in \mathbb{N}$
`MMLEARN`	MM (1.2.2)	-
`PARZENLEARN`	Parzen (1.2.1)	$h > 0$
`QDALEARN`	QDA (1.2.1)	$\lambda > 0$
`RRLEARN`	RR (1.2.2)	$\lambda > 0$
`RVFLLEARN`	RVFL (1.2.3)	$J \in \mathbb{N}, \lambda > 0$

Firstly, we present a summary of supporting and learning functions. We use two basic functions for the preprocessing, the RAO method preprocessing and data whitening via universal function for different sizes of dimensionality reductions. The basic functions are presented in Table 7.1 together with the main changing parameters.

We have set special functions for optimal unioning of the hidden classes. These functions are summarized in Table 7.2.

Table 7.2: Functions for Optimal Union of Hidden Classes

Name	Description
`OUHC_ACC_MAX`	Accuracy maximization
`OUHC_SESTAR_MAX`	Critical sensitivity maximization
`HIDDEN_CLASSES`	Hidden classes formation
`HIDDEN_LEARN`	Learning of hidden classes unioning
`HIDDEN_RESP`	Classification by hidden classes

Thirdly, we use special function for different validation techniques. In table 7.3 is presented a summary of functions used for the validation and quality evaluation of proposed classifier. The classifier proposes naive, m-fold as well as the Leave-One-Out validation technique. The final classifier characteristics are based on the relevant contingency table.

Table 7.3: Functions for Validation

Name	Description
`CROSS_NAIVE`	Naive validation
`CROSS_MFOLD`	M-fold cross-validation
`CROSS_L1OUT`	Leave-One-Out cross-validation
`CROSS_STAT`	Evaluation of accuracy and sensitivities

Conclusions

We have presented the novel imperfect approach to classification tasks. To provide a comprehensive overview of the current state of the art in classification methods, we conducted a thorough review of existing literature and identified a range of traditional classification methods and approaches for the data clustering. To evaluate the effectiveness of our approach, we conducted a series of rigorous experiments, using a range of data sets and comparing our approach with the selected traditional methods. Our results demonstrate that the novel approach is able to outperform the traditional methods.

In the novel theory of imperfect learning, we build on the critical sensitivity as the main benchmark for the classifier assessment. The reason is not to omit the classes with a low number of representatives because we focus on the lowest class accuracy among all the classes as a benchmark for the evaluation of classification. There is usually no previous knowledge regarding the different importance of the individual classes, therefore the critical sensitivity has no prejudices and guarantees equal importance of all the classes. We have formulated the imperfect learning strategy using the linear programming theory.

In a separate chapter, we have proposed novel methods for the self-organization derived from the traditional Kohonen learning strategy. Our approach is based on the idea that diffusion is a fundamental process in nature, occurring in everything from chemical reactions to the movement of particles in a fluid. By incorporating the diffusion into the learning process, we are able to create a more dynamic and flexible approach to the self-organization. We have developed two novel learning strategies based on the diffusion - the traditional diffusion and the anomalous diffusion - both of which demonstrate improvements or at least comparability in the most cases over the traditional Kohonen learning. Such self-organization proposed new ways of explanation and it could serve as a valuable alternative to the Kohonen learning.

We have presented the novel theory of hidden classes and their unioning. These hidden classes can be formed in several ways. The traditional imperfect supervised learning strategies can be used as well as the unsupervised learning techniques. There is no need for perfect classification for the formation of the hidden classes. Therefore, it is not a hard task to generate a higher number of the hidden classes. The imperfectness can be led by selecting only a limited number of features or patterns. It is crucial to obtain a higher number of the classes compared to the original classification task. The core theory of the new approach is

the optimal unioning of hidden classes.

Our novel multi-classifier has a five-layer structure. The first layer is for the optional nonlinear transformation, the second layer is for the optional linear transformation. The third layer is formed by the imperfect classifiers which lead to so-called hidden classifiers, and in the last layer, there is the final classifier. The theory of optimal unioning based on the linear programming theory is the core part of the final classification. The proposed classifier is flexible and the user is able to use individual settings.

In the experimental part, we provided a detailed analysis of the novel classifier in ten different traditional datasets. We have compared the results with the traditional perfect learning strategies. The novel classifier is able to over-perform the perfect approach in most cases. Additionally, we have presented the results on two specific real datasets, where the competitiveness of the novel classifier was confirmed.

The classifier engine is built in Matlab and the basic structure and its organization to elementary functions are briefly summarized in the last chapter. The selected source code examples are presented in the annex.

Bibliography

[1] R. O. Duda, P. E. Hart, D. G. Stork, Pattern classification, John Wiley & Sons, 2012.

[2] S. J. Russell, P. Norvig, Artificial Intelligence: a modern approach, 3rd Edition, Pearson, 2009.

[3] G. J. McLachlan, Discriminant analysis and statistical pattern recognition, Vol. 544, John Wiley & Sons, 2004.

[4] T. Cacoullos, Discriminant Analysis and Applications, Elsevier Science, 2014.

[5] E. Parzen, On estimation of a probability density function and mode, The annals of mathematical statistics 33 (3) (1962) 1065–1076.

[6] S. DW, Multivariate density estimation: Theory, practice, and visualization (1992).

[7] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman & Hall, London, 1986.

[8] D. Loftsgaarden, A nonparametric density function/do loftsgaarden cp queensberry, Ann. Math. Stat 36 (1965) 1049–1051.

[9] B. Silverman, Density Estimation for Statistics and Data Analysis, CRC Press, 2018.

[10] M. J. Zaki, W. Meira Jr, W. Meira, Data mining and analysis: fundamental concepts and algorithms, Cambridge University Press, 2014.

[11] B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, COLT '92, New York, NY, USA, 1992, pp. 144–152.

[12] A. E. Hoerl, R. W. Kennard, Ridge regression: Biased estimation for nonorthogonal problems, Technometrics 12 (1) (1970) 55–67.

[13] D. W. Marquardt, R. D. Snee, Ridge regression in practice, The American Statistician 29 (1) (1975) 3–20.

[14] V. Vovk, Kernel ridge regression, in: Empirical inference, Springer, 2013, pp. 105–116.

[15] S. An, W. Liu, S. Venkatesh, Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression, Pattern Recognition 40 (8) (2007) 2154–2162.

[16] D. Husmeier, D. Husmeier, Random vector functional link (rvfl) networks, Neural Networks for Conditional Probability Estimation: Forecasting Beyond Point Predictions (1999) 87–97.

[17] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems 2 (4) (1989) 303–314.

[18] S. Mirjalili, How effective is the grey wolf optimizer in training multi-layer perceptrons, Applied Intelligence 43 (1) (2015) 150–161.

[19] S. Chen, C. F. Cowan, P. M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Transactions on neural networks 2 (2) (1991) 302–309.

[20] G. Foody, Supervised image classification by mlp and rbf neural networks with and without an exhaustively defined set of classes, International Journal of Remote Sensing 25 (15) (2004) 3091–3104.

[21] C. Karlsson, Handbook of research on cluster theory, Vol. 1, Edward Elgar Publishing, 2010.

[22] N. Kambhatla, T. K. Leen, Dimension reduction by local principal component analysis, Neural computation 9 (7) (1997) 1493–1516.

[23] R. Sibson, Slink: an optimally efficient algorithm for the single-link cluster method, The computer journal 16 (1) (1973) 30–34.

[24] R. Sibson, SLINK: An optimally efficient algorithm for the single-link cluster method, The Computer Journal 16 (1) (1973) 30–34.

[25] P. Goyal, S. Kumari, S. Sharma, D. Kumar, V. Kishore, S. Balasubramaniam, N. Goyal, A fast, scalable slink algorithm for commodity cluster computing exploiting spatial locality, in: 2016 IEEE 18th HPCC; IEEE 14th SmartCity; IEEE 2nd DSS, IEEE, 2016, pp. 268–275.

[26] M. Schmidt, A. Kutzner, K. Heese, A novel specialized single-linkage clustering algorithm for taxonomically ordered data, Journal of theoretical biology 427 (2017) 1–7.

[27] D. Defays, An efficient algorithm for a complete link method, The Computer Journal 20 (4) (1977) 364–366.

[28] D. Krznaric, C. Levcopoulos, Fast algorithms for complete linkage clustering, Discrete & Computational Geometry 19 (1) (1998) 131–145.

[29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise., in: KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231.

[30] N. Antony, A. Deshpande, Domain-driven density based clustering algorithm, in: Proceedings of International Conference on ICT for Sustainable Development, Springer, 2016, pp. 705–714.

[31] L. Bai, X. Cheng, J. Liang, H. Shen, Y. Guo, Fast density clustering strategies based on the k-means algorithm, Pattern Recognition 71 (2017) 375–386.

[32] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, L. Shao, Real-time superpixel segmentation by dbscan clustering algorithm, IEEE Transactions on Image Processing 25 (12) (2016) 5933–5942.

[33] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu, Dbscan revisited, revisited: why and how you should (still) use dbscan, ACM Transactions on Database Systems (TODS) 42 (3) (2017) 19.

[34] T. Kohonen, Self-organized formation of topologically correct feature maps, Biological Cybernetics 43 (1) (1982) 59–69.

[35] A. Lavecchia, Machine-learning approaches in drug discovery: methods and applications, Drug Discovery Today 20 (3) (2015) 318 – 331.

[36] O. A. Moldes, J. C. Mejuto, R. Rial-Otero, J. Simal-Gandara, A critical review on the applications of artificial neural networks in winemaking technology, Critical Reviews in Food Science and Nutrition 57 (13) (2017) 2896–2908.

[37] J. A. F. Costa, A. P. V. Pinto, J. R. de Andrade, M. G. de Medeiros, Clustering of regional hdi data using self-organizing maps, in: 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI), 2017, pp. 1–6.

[38] A. Urmos, Z. Farkas, M. Farkas, T. Sandor, L. T. Koczy, A. Nemcsics, Fuzzy and kohonen som based classification of different 0d nanostructures, in: 2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI), 2017, pp. 365–370.

[39] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, Springer Berlin Heidelberg, 2012.

[40] E. Alonso, Computational Neuroscience for Advancing Artificial Intelligence: Models, Methods and Applications: Models, Methods and Applications, Premier reference source, Medical Information Science Reference, 2010.

[41] I. Jolliffe, Principal component analysis, Springer, 2011.

[42] Y. C. Eldar, A. V. Oppenheim, Mmse whitening and subspace whitening, IEEE Transactions on Information Theory 49 (7) (2003) 1846–1851.

[43] L. Huang, Y. Zhou, F. Zhu, L. Liu, L. Shao, Iterative normalization: Beyond standardization towards efficient whitening, in: Proceedings of the ieee/cvf conference on computer vision and pattern recognition, 2019, pp. 4874–4883.

[44] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K.-R. Mullers, Fisher discriminant analysis with kernels, in: Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468), Ieee, 1999, pp. 41–48.

[45] C. Croux, P. Filzmoser, K. Joossens, Classification efficiencies for robust linear discriminant analysis, Statistica Sinica (2008) 581–599.

[46] C. R. Rao, H. Toutenburg, Linear models, in: Linear models, Springer, 1995, pp. 3–18.

[47] T.-T. Wong, Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation, Pattern Recognition 48 (9) (2015) 2839–2846.

[48] Q. F. Gronau, E.-J. Wagenmakers, Limitations of bayesian leave-one-out cross-validation for model selection, Computational brain & behavior 2 (1) (2019) 1–11.

[49] E. W. Steyerberg, Validation of prediction models, in: Clinical prediction models, Springer, 2019, pp. 329–344.

[50] L. Xu, H.-Y. Fu, M. Goodarzi, C.-B. Cai, Q.-B. Yin, Y. Wu, B.-C. Tang, Y.-B. She, Stochastic cross validation, Chemometrics and Intelligent Laboratory Systems 175 (2018) 74–81.

[51] G. Pölzlbauer, Survey and comparison of quality measures for self-organizing maps (2004).

[52] L. Hamel, Som quality measures: An efficient statistical approach, in: Proceedings of the 11th International Workshop WSOM 2016, Springer, Houston, 2016, pp. 49–59.

[53] D. Graupe, Deep learning neural networks: Design and case studies, World Scientific Publishing Company, 2016.

[54] E. Oja, S. Kaski, Kohonen Maps, Elsevier Science, 1999.

[55] W. L. Chang, L. M. Pang, K. M. Tay, Application of self-organizing map to failure modes and effects analysis methodology, Neurocomputing 249 (2017) 314 – 320.

[56] R. Hrebik, J. Kukal, Diffusion modelling: Topographic error of som under control, Neural Processing Letters 54 (2) (2022) 835–852.

[57] E. Cussler, Diffusion: Mass Transfer in Fluid Systems, Cambridge Series in Chemical Engineering, Cambridge University Press, 2009.

[58] J. Espenson, Chemical Kinetics and Reaction Mechanisms, Advanced Chemistry Series, McGraw-Hill, 1995.

[59] S. Kornblith, R. Q. Quiroga, C. Koch, I. Fried, F. Mormann, Persistent single-neuron activity during working memory in the human medial temporal lobe, Current Biology 27 (7) (2017) 1026 – 1032.

[60] Y. Yun, The moments of a diffusion process, Statistics & Probability Letters 138 (2018) 36 – 41.

[61] D. Brogioli, A. Vailati, Diffusive mass transfer by nonequilibrium fluctuations: Fick's law revisited, Phys. Rev. E 63 (2000) 012105.

[62] M. Senapati, Advanced Engineering Chemistry, Laxmi Publications, 2006.

[63] C. Pozrikidis, The Fractional Laplacian, CRC Press, 2016.

[64] S. Kotz, S. Nadarajah, Multivariate T-Distributions and Their Applications, Cambridge University Press, 2004.

[65] B. Davies, Integral Transforms and Their Applications, Texts in Applied Mathematics, Springer New York, 2002.

[66] R. Hrebik, J. Kukal, Anomalous and traditional diffusion modelling in som learning, Archives of Control Sciences 29 (2019).

[67] E. Kuriscak, P. Marsalek, J. Stroffek, P. G. Toth, Biological context of hebb learning in artificial neural networks, a review, Neurocomputing 152 (2015) 27–35.

[68] G.-B. Huang, What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle, Cognitive Computation 7 (2015) 263–278.

[69] D. D. Lewis, R. E. Schapire, J. P. Callan, R. Papka, Training algorithms for linear text classifiers, in: Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, 1996, pp. 298–306.

[70] R. Hrebik, J. Kukal, J. Jablonsky, Optimal unions of hidden classes, Central European Journal of Operations Research 27 (1) (2019) 161–177.

[71] R. Hrebik, J. Kukal, Concept of hidden classes in pattern classification, Artificial Intelligence Review (2023) 1–18.

[72] R. M. Sakia, The box-cox transformation technique: a review, Journal of the Royal Statistical Society: Series D (The Statistician) 41 (2) (1992) 169–178.

[73] D. Dua, C. Graff, UCI machine learning repository (2021).
URL `http://archive.ics.uci.edu/ml`

[74] O. L. Mangasarian, W. H. Wolberg, Cancer diagnosis via linear programming, Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (1990).

[75] F. Khozeimeh, R. Alizadehsani, M. Roshanzamir, A. Khosravi, P. Layegh, S. Nahavandi, An expert system for selecting wart treatment method, Computers in biology and medicine 81 (2017) 167–175.

[76] F. Khozeimeh, F. Jabbari Azad, Y. Mahboubi Oskouei, M. Jafari, S. Tehranian, R. Alizadehsani, P. Layegh, Intralesional immunotherapy compared to cryotherapy in the treatment of warts, International journal of dermatology 56 (4) (2017) 474–478.

[77] M. Patricio, J. Pereira, J. Crisostomo, P. Matafome, M. Gomes, R. Seicca, F. Caramelo, Using resistin, glucose, age and bmi to predict the presence of breast cancer, BMC cancer 18 (1) (2018) 29.

[78] I.-C. Yeh, K.-J. Yang, T.-M. Ting, Knowledge discovery on rfm model using bernoulli sequence, Expert Systems with Applications 36 (3) (2009) 5866–5871.

[79] J. Kukal, K. Horaisova, Could k-nn classifier be useful in tree leaves recognition?, Archives of Control Sciences (2014).

[80] K. Horaisova, J. Kukal, Leaf classification from binary image via artificial intelligence, Biosystems Engineering 142 (2016) 83–100.

Appendix A

Source Code Examples

A.1 Data Transforming

For the optional data preprocessing, we present source codes of two basic functions used for data transformation. Firstly, there is shown the code for the RAO method and secondly, the code used when data whitening is required.

The RAO method source code:

```
function [ex,W,Y]=RAO(X,ystar)
% RAO discriminant analysis
H=max(ystar);
m=size(X,1);
ex=mean(X,1);
for k=1:m
    X(k,:)=X(k,:)-ex;
end
ST=X'*X;
SW=0*ST;
for k=1:H
    B=X(ystar==k,:);
    m=size(B,1);
    bx=mean(B,1);
    for j=1:m
        B(j,:)=B(j,:)-bx;
    end
    SW=SW+B'*B;
end
SB=ST-SW;
M=pinv(SW)*SB;
M=(M+M')/2;
[W,L]=eig(M);
lambda=diag(L);
```

```
[lambda,ind]=sort(lambda,'descend');
lambda=lambda(1:H-1);
ind=ind(1:H-1);
W=W(:,ind);
W=W(:,lambda>0);
n=size(W,2);
for k=1:n
    [~,ind]=max(abs(W(:,k)));
    ind=ind(1);
    W(:,k)=W(:,k)*sign(W(ind,k));
end
Y=X*W;
ex=ex';
```

The DWH source code:

```
function [ex,W,Y,lambda]=DWH(X,D)
[m,n]=size(X);
ex=mean(X,1);
for k=1:m
    X(k,:)=X(k,:)-ex;
end
if m<n
    A=X*X'/sqrt(m-1);
else
    A=X'*X/(m-1);
end
[E,LAMBDA]=eig(A);
if m<n
    W=zeros(m,D);
    for k=1:D
        W(:,k)=E(:,end-k+1)/max(LAMBDA(end-k+1,end-k+1),1e-100);
    end
    W=X'*W;
else
    W=zeros(n,D);
    for k=1:D
        W(:,k)=E(:,end-k+1)/max(LAMBDA(end-k+1,end-k+1),1e-100)^(1/2);
    end
end
lambda=diag(LAMBDA);
lambda=cumsum(lambda(end:-1:1));
lambda=100*lambda(1:D)'/lambda(end);
Y=X*W;
ex=ex';
```

A.2 Learning and Classification Functions

In this section, we present four selected methods used for learning itself and responses. The methods were selected based on the results in Section 6.3 for the KRR, the LDA, the QDA, and the RVFL methods.

A.2.1 KRR

The source code of the learning function for the KRR method is as follows.

```
function resplearn=KRRLEARN(X,ystar,par)
sigma=par(1);
lambda=par(2);
m=size(X,1);
N=max(ystar);
Y=zeros(m,N);
for k=1:m
   Y(k,ystar(k))=1;
end
G=ones(m,m);
for i=1:m-1
    for j=i+1:m
        G(i,j)=exp(-norm(X(i,:)-X(j,:))^2/2/sigma^2);
        G(j,i)=G(i,j);
    end
end
A=pinv(G+lambda*eye(m,m))*Y;
resplearn={A,X,sigma};
```

The source code of the function generating response for the KRR method is as follows.

```
function y=KRRRESP(x,resplearn)
A=resplearn{1};
X=resplearn{2};
sigma=resplearn{3};
m=size(X,1);
ke=ones(m,1);
for i=1:m
    ke(i)=exp(-norm(X(i,:)-x')^2/2/sigma^2);
end
y=A'*ke;
[~,ind]=max(y);
if length(ind)==1
    y=ind;
```

```
else
    y=0;
end
```

A.2.2 LDA

The source code of the learning function for the LDA method is as follows.

```
function resplearn=LDALEARN(X,ystar,par)
lambda=par(1);
[m,n]=size(X);
N=max(ystar);
MU=zeros(N,n);
SIGMA=zeros(n,n);
for k=1:N
   ind=find(ystar==k);
   mpatt=length(ind);
   if isempty(ind)
       MU(k,:)=inf;
       %SIGMA(:,:,k)=0;
       continue
   end
   B=X(ind,:);
   MU(k,:)=mean(B,1);
   sss=cov(B)+lambda*eye(n,n);
   SIGMA=SIGMA+sss*mpatt;
end
SIGMA=SIGMA/m;
SIGMAINV=pinv(SIGMA);
logdet=log(det(SIGMA));
resplearn={MU,SIGMAINV,logdet};
```

The source code of the function generating response for the LDA method is as follows.

```
function y=LDARESP(x,resplearn)
MU=resplearn{1};
SIGMAINV=resplearn{2};
logdet=resplearn{3};
N=size(MU,1);
d=zeros(N,1);
for k=1:N
    mu=MU(k,:);
    d(k)=logdet+(x-mu')'*SIGMAINV*(x-mu');
end
[~,ind]=min(d);
```

```
if length(ind)==1
    y=ind;
else
    y=0;
end
```

A.2.3 QDA

The source code of the learning function for the QDA method is as follows.

```
function resplearn=QDALEARN(X,ystar,par)
lambda=par(1);
n=size(X,2);
N=max(ystar);
MU=zeros(N,n);
SIGMAINV=zeros(n,n,N);
logdet=zeros(N,1);
for k=1:N
   ind=find(ystar==k);
   if isempty(ind)
       MU(k,:)=inf;
       SIGMAINV(:,:,k)=inf;
       logdet(k)=inf;
       continue
   end
   B=X(ind,:);
   MU(k,:)=mean(B,1);
   SIGMA=cov(B)+lambda*eye(n,n);
   SIGMAINV(:,:,k)=pinv(SIGMA);
   logdet(k)=log(det(SIGMA));
end
resplearn={MU,SIGMAINV,logdet};
```

The source code of the function generating response for the QDA method is as follows.

```
function y=QDARESP(x,resplearn)
MU=resplearn{1};
SIGMAINV=resplearn{2};
logdet=resplearn{3};
N=length(logdet);
d=zeros(N,1);
for k=1:N
    mu=MU(k,:);
    d(k)=logdet(k)+(x-mu')'*SIGMAINV(:,:,k)*(x-mu');
end
```

```
[~,ind]=min(d);
if length(ind)==1
    y=ind;
else
    y=0;
end
```

A.2.4 RVFL

The source code of the learning function for the RVFL method is as follows.

```
function resplearn=RVFLLEARN(X,ystar,par)
J=par(1);
qmax=par(2);
q0max=par(3);
lambda=par(4);
[m,n]=size(X);
N=max(ystar);
Y=zeros(m,N);
for k=1:m
   Y(k,ystar(k))=1;
end
Q=[q0max*unifrnd(-1,1,1,J);
qmax*unifrnd(-1,1,n,J)];
XNEW=[ones(m,1),X];
G=tanh(XNEW*Q);
REG=eye(n+J+1,n+J+1);
REG(1,1)=0;
XNEW=[XNEW,G];
W=pinv(XNEW'*XNEW+lambda*REG)*XNEW'*Y;
resplearn={Q,W};
```

The source code of the function generating response for the RVFL method is as follows.

```
function y=RVFLRESP(x,resplearn)
Q=resplearn{1};
W=resplearn{2};
x=[1;x];
g=Q'*x;
x=[x;g];
y=W'*x;
[~,ind]=max(y);
```

```
if length(ind)==1
    y=ind;
else
    y=0;
end
```

A.3 General Source Code Examples

In this section, we first present the functions used for hidden learning and hidden classification. Then we show the source code of Leave-One-Out and M-fold validations. Finally, we present the accuracy and sensitivity evaluation function.

The source code of hidden learning function is as follows:

```
function [redresp,C,r]=HIDDEN_LEARN(X,ystar)
global ssemin
resp=MULTI_LEARN(X,ystar);
YCOL=FORALL_RESP(X,resp);
[c,C,redresp]=HIDDEN_CLASSES(resp,YCOL);
mixed=0;
F=OUHC_CONTING(ystar,c);
if ssemin<=0
    [~,~,semin]=OUHC_SESTAR_MAX(F,mixed);
end
r=OUHC_ACC_MAX(F,semin,mixed);
```

The source code of hidden classification function is as follows:

```
function [c,C,redresp,Y]=HIDDEN_CLASSES(resp,YCOL)
[Y,ind]=unique(YCOL',’rows’);Y=Y';
redresp=resp(ind);
[C,~,c]=unique(Y,’rows’);
```

The Leave-One-Out validation technique is represented as follows:

```
function y=CROSS_L1OUT(X,ystar)
y=0*ystar;m=length(ystar);
for k=1:m
   BX=[X(1:k-1,:);X(k+1:m,:)];
   by=[ystar(1:k-1);ystar(k+1:m)];
   sx=X(k,:);
   [redresp,C,r]=HIDDEN_LEARN(BX,by);
   y(k)=HIDDEN_RESP(sx',redresp,C,r);
end
```

The M-fold validation technique is represented as follows:

```
function y=CROSS_MFOLD(X,ystar,M)
y=0*ystar;m=length(ystar);
H=floor(m/M);
res=mod(m,M);
for k=1:res
    level=(k-1)*(H+1)+1;
    BX=[X(1:level-1,:);X(level+H+1:m,:)];
    by=[ystar(1:level-1);ystar(level+H+1:m)];
    [redresp,C,r]=HIDDEN_LEARN(BX,by);
    for kkk=level:level+H
        sx=X(kkk,:);
        y(kkk)=HIDDEN_RESP(sx',redresp,C,r);
    end
end
for k=1:M-res
    level=(k-1)*H+1+(H+1)*res;
    BX=[X(1:level-1,:);X(level+H:m,:)];
    by=[ystar(1:level-1);ystar(level+H:m)];
    [redresp,C,r]=HIDDEN_LEARN(BX,by);
    for kkk=level:level+H-1
        sx=X(kkk,:);
        y(kkk)=HIDDEN_RESP(sx',redresp,C,r);
    end
end
```

Basic characteristics for classifier assessment are calculated using two following functions:

```
function [acc,ase,sestar,se]=CROSS_STAT(F)
m=sum(sum(F));
acc=sum(diag(F))/m;
N=size(F,1);
se=zeros(N,1);
for k=1:N
    s=sum(F(k,:));
    if s>0
        se(k)=F(k,k)/s;
    else
        se(k)=1; %optimism
    end
end
ase=mean(se);
sestar=min(se);
```

```
function F=CROSS_CONTING(ystar,y)
m=length(ystar);
N=max(ystar);
H=N+1;
F=zeros(N,H);
for k=1:m
    i=ystar(k);
    j=y(k);
    if j==0
        j=H;
    end
    F(i,j)=F(i,j)+1;
end
```

Appendix B

Selected Publications in the Impact Journals

R. Hrebik, J. Kukal, Concept of hidden classes in pattern classification, *Artificial Intelligence Review*, 2023, 1-18.

R. Hrebik, J. Kukal, Diffusion modelling: Topographic error of som under control, *Neural Processing Letters*, 2022, 54.2: 835-852.

R. Hrebik, J. Kukal, Optimal unions of hidden classes, *Central European Journal of Operations Research*, 27 (1) (2019) 161–177.

Artificial Intelligence Review
https://doi.org/10.1007/s10462-023-10430-6

Concept of hidden classes in pattern classification

Radek Hrebik[1] · **Jaromir Kukal**[1]

Accepted: 9 February 2023

Abstract
Our paper presents a novel approach to pattern classification. The general disadvantage of a traditional classifier is in too different behaviour and optimal parameter settings during training on a given pattern set and the following cross-validation. We describe the term critical sensitivity, which means the lowest reached sensitivity for an individual class. This approach ensures a uniform classification quality for individual class classification. Therefore, it prevents outlier classes with terrible results. We focus on the evaluation of critical sensitivity, as a quality criterion. Our proposed classifier eliminates this disadvantage in many cases. Our aim is to present that easily formed hidden classes can significantly contribute to improving the quality of a classifier. Therefore, we decided to propose classifier will have a relatively simple structure. The proposed classifier structure consists of three layers. The first is linear, used for dimensionality reduction. The second layer serves for clustering and forms hidden classes. The third one is the output layer for optimal cluster unioning. For verification of the proposed system results, we use standard datasets. Cross-validation performed on standard datasets showed that our critical sensitivity-based classifier provides comparable sensitivity to reference classifiers.

Keywords Pattern classification · Data clustering · Dimensionality reduction · Hidden classes · Multi-classifier

List of Symbols

Sets

$\mathscr{S}$ Pattern set
$\mathbb{N}$ Natural numbers
$\mathbb{R}$ Real numbers
$\mathscr{C}$ Output class
$\mathscr{H}$ Hidden class

✉ Radek Hrebik
Radek.Hrebik@fjfi.cvut.cz

Jaromir Kukal
Jaromir.Kukal@fjfi.cvut.cz

[1] FNSPE, CTU in Prague, Trojanova 13, Prague 12000, Prague, Czech Republic

Published online: 21 February 2023

Variables

n	Number of features
m	Number of patterns
H	Number of hidden classes
N	Number of classes
D	Reduced dimension
$\mathbf{p}$	Pattern vector
$\mathbf{X}$	Input matrix
$\mathbf{y}$	Classifier output vector
$\mathbf{y}^*$	Desired output vector
y	Classifier output/response
y^*	Desired output value
$\mathbf{w}$	Weight vector
w_0	Bias

Functions

c	Classifier

1 Introduction

There exist a lot of different classifiers designed for classification from n-dimensional space. These classifiers are usually used for classification in two or more classes and are based on various principles. For example, linear combination (Orozco-Alzate et al. 2019), use of mutual distances (Shahid and Singh 2019; Liu and Wang 2022; Veneri et al. 2022), variants of gradient boosting (Bentejac et al. 2021; Liang and Sur 2022), adaboost (Hu et al. 2020), and bagging (Medina-Pérez et al. 2017; Jafarzadeh et al. 2021).

Our new approach is based on an assumption that for data with a complex structure, it is beneficial to divide the data into an unspecified number of sub-groups. These artificial groups, so-called hidden classes, can be formed easily using unsupervised or supervised learning techniques. We can use small low-level classifiers with limited validity and focus on data belonging to these classes. The way how we form hidden classes is not crucial, but they could be helpful for the final classification, which will be improved subsequently by their optimal unification. Such an optimal union leads to better classification results than using only the original datasets. This idea is the cornerstone of the proposed classifier. Furthermore, it is necessary to emphasize that we focus on the sensitivity of each, individual class, and not omit any of them.

We propose using simple clustering methods resulting in more clusters than the final classes, which will not be many or few. After forming such clusters, we propose a union of them with the optimal union method (Hrebik et al. 2019). Such unioning enables the formation of a classifier with the highest possible critical sensitivity. Our aim is to introduce a novel classifier suitable for processing data with a higher dimension, so dimension reduction must also be part of our classifier. Dimensionality reduction can be optional, but in our opinion, it generally increases the efficiency of the system and our recommendation is data whitening, thanks to which we get a dimensionless description.

A separate part of our paper considers meaning of critical sensitivity and why we propagate this criterion. The basic idea is that in the case of classification into classes, we have a separately evaluated percentage of correctly classified patterns for each class, which we refer

to as the sensitivity of the relevant class. However, most classifiers maximize their accuracy on all data, which, of course, represents an unbalanced evaluation. We focus on the class providing worst results, and this sensitivity we note as critical sensitivity. Even simple clustering methods, such as DBSCAN, can provide high-quality clusters, which by unification will create a classifier in which even the worst class will have sufficient level sensitivity. Therefore, even in the case of an unbalanced size of classes, using our proposed classifier, we do not omit even the smallest one because of a low number of patterns. Our proposed method focuses on generating hidden classes from current datasets. Some authors suggest to generate synthetic samples (Rekha and Madhu 2022).

In this section, we provide a brief basic summary of the current state of research. Pattern recognition (Duda et al. 2012) represents a common aim of artificial intelligence and machine learning. In the case of machine learning, we distinguish between unsupervised and supervised approaches. The typical clustering models (Karlsson 2010) are connectivity, centroid, distribution, density, subspace, graph-based, and neural network ones (Back et al. 2018; Shi et al. 2019; Lin et al. 2020). A well-known unsupervised neural network model is represented by self-organizing maps (SOM). We can include subspace models as Principal Component Analysis or Independent Component Analysis as data processing techniques for dimension reduction (Eldar and Oppenheim 2003; Jolliffe 2011; Nguyen and Holmes 2019) of an original dataset consisting of a large number of interrelated variables. Dimensionality reduction leads to data representation using fewer features. Another approach to dimension reduction represent linear discriminant analysis into two classes (Eldar and Oppenheim 2003; Croux et al. 2008) or more classes (Rao and Toutenburg 1995).

Our proposed approach consists of three steps:

- Dimensionality reduction and standardization,
- Data clustering in the space of reduced dimension,
- Hidden class forming and their optimal union to desired output classes.

Dimensionality reduction is based on data whitening (Eldar and Oppenheim 2003) or multiclass discriminant analysis, as an alternative, in the first step. The following clustering is performed by parameter-driven DBSCAN (Ester et al. 1996) generating several classes, their structure, and outliers. Proposed clustering technique is quick and can be reduced to the SLINK (Sibson 1973) algorithm in many cases. In the last step, we define hidden classes as clusters from the previous step. The relationship between these hidden classes and output classes learned is optimized using a binary programming technique that is focused on the maximization of classifier critical sensitivity.

We summarize several principles of multi-classifiers in the second section focusing on data whitening, multiple discriminant analysis, data clustering, and the concept of hidden classes. We discuss the framework and structure of a novel multi-classifier in the third section. The numerical experiments on basic pattern sets and resulting optimal settings and quality measures are summarized in the next section. The last section concludes our proposed classifier.

2 Multi-classification preliminaries

Basic facts related to the classification into several classes are summarized in this section. The optional methods of preprocessing are discussed first as unsupervised and supervised approaches. Data clustering is then discussed as a method of how to generate hidden

classes. Finally, the concept of unioning of hidden classes is presented as a kernel of the novel approach.

2.1 Multi-classification Task

Basic frame of classification of vector patterns (Duda et al. 2012) into several classes is established first. Let $n, m, N \in \mathbb{N}$ be number of features, patterns, and classes satisfying $N \geq 2$. Let $\mathbf{x} \in \mathbb{R}^n$ be the feature vector and $y, y^* \in \{1, ..., N\}$ be a classifier output and its required value, denoting a pattern as $\mathbf{p} = (\mathbf{x}, y^*)$, the classifier is defined as a function

$$\mathrm{c} : \mathbb{R}^n \to \{1, ..., N\}$$

and the classifier response is therefore $y = \mathrm{c}(\mathbf{x})$. Denoting $\mathbf{x}_k \in \mathbb{R}^n$, $y_k^* \in \{1, ..., N\}$ as a feature vector and given output of k-th pattern, we define a pattern set as

$$\mathscr{S} = \{(\mathbf{x}_k, y_k^*) : k = 1, ..., m\}$$

The pattern set can be represented by any input matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ and any output vector $\mathbf{y}^* \in \{1, ..., N\}^m$. Any classifier is a complex system that applies various data processing techniques to obtain the final decision. Selected approaches are summarised in the following subsections.

2.2 Data whitening

The first but optional step of any classification is an efficient transformation that decreases the number of features but saves information about pattern differences. The main idea of principal component analysis (PCA) (Jolliffe 2011) is to reduce the dimensionality of the original dataset consisting of a large number of interrelated variables. The reduction retains as much as possible of the variation present in the data set. The aim is achieved by transforming into a new set of variables called principal components. These components are uncorrelated and ordered so that the first few retain most of the variation present in all of the original variables.

Let $D \in \mathbb{N}$ be reduced dimension satisfying $D < n$. The dimensionality reduction from $\mathbb{R}^n$ to $\mathbb{R}^D$ using PCA is based on a linear transformation

$$\mathbf{z} = \mathrm{PCA}(\mathbf{x}) = \mathbf{W}_1^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0) \tag{1}$$

The PCA is designed to satisfy $\mathrm{E}\,\mathbf{z} = \mathbf{0}$ and $\mathrm{var}\,\mathbf{z} = \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix. Resulting parameters of PCA are $\mathbf{W}_1 \in \mathbb{R}^{n \times D}$ and

$$\mathbf{x}_0 = \frac{1}{m} \sum_{k=1}^{m} \mathbf{x}_k \tag{2}$$

The transforming matrix $\mathbf{W}_1$ is calculated as follows. First, we shift the input matrix to obtain $\mathbf{X}_{\mathrm{S}} = \mathbf{X} - \mathbf{1}_m \mathbf{x}_0^{\mathrm{T}}$, where $\mathbf{1}_m$ is m-dimensional vector of units. Then we calculate a covariance matrix $\mathbf{A} = \mathbf{X}_{\mathrm{S}}^{\mathrm{T}} \mathbf{X}_{\mathrm{S}} \geq 0$ and apply Eigen-Value Decomposition (EVD) as finding of eigenvalues $\mathbf{v} \in \mathbb{R}^n$ and eigenvectors $\lambda \geq 0$ in equation $(\mathbf{A} - \lambda \mathbf{I}_n)\mathbf{v} = \mathbf{0}$ where $\mathbf{I}_n \in \mathbb{R}^{n \times n}$ is identity matrix.

The solutions can be ordered as $\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(D)} \geq 0$ with corresponding normalized eigenvectors $\mathbf{v}_{(1)}, \mathbf{v}_{(2)}, ..., \mathbf{v}_{(D)}$. Resulting PCA matrix (Jolliffe 2011) is

$$\mathbf{W}_1 = (\mathbf{v}_{(1)}, \mathbf{v}_{(2)}, ..., \mathbf{v}_{(D)}) \tag{3}$$

and the dimensionality reduction generates new feature matrix $\mathbf{Z} = \mathbf{X}_S\mathbf{W}_1$.

Data Whitening (DWH) (Eldar and Oppenheim 2003) represents improved process of PCA, which guarantees unit covariance matrix of resulting vector. The transform is defined as

$$\mathbf{z} = \mathrm{DWH}(\mathbf{x}) = \mathbf{W}_2^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0) \tag{4}$$

The matrix $\mathbf{W}_2^{\mathrm{T}}$ is designed to satisfy $\mathrm{E}\,\mathbf{z} = \mathbf{0}$ and $\mathrm{var}\,\mathbf{z} = \mathbb{I}_n$. Using the result of EVD we directly calculate (Eldar and Oppenheim 2003)

$$\mathbf{W}_2 = \left(\frac{\mathbf{v}_{(1)}}{\sqrt{\lambda_{(1)}}}, \frac{\mathbf{v}_{(2)}}{\sqrt{\lambda_{(2)}}}, ..., \frac{\mathbf{v}_{(D)}}{\sqrt{\lambda_{(D)}}} \right) \tag{5}$$

Due to duality, we can perform the data whitening for $m < n$ in more efficient way. We calculate $\mathbf{B} = \mathbf{X}_S\mathbf{X}_S^{\mathrm{T}} \geq 0$ and perform its EVD. Resulting EVD equation is $(\mathbf{B} - \lambda\mathbf{I}_m)\mathbf{u} = \mathbf{0}$. The solutions can be ordered again as $\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(D)} \geq 0$ with corresponding normalized eigenvectors $\mathbf{u}_{(1)}, \mathbf{u}_{(2)}, ..., \mathbf{u}_{(D)} \in \mathrm{R}^m$. Resulting whitening matrix is

$$\mathbf{W}_2 = \mathbf{X}_S^{\mathrm{T}} \left(\frac{\mathbf{u}_{(1)}}{\lambda_{(1)}}, \frac{\mathbf{u}_{(2)}}{\lambda_{(2)}}, ..., \frac{\mathbf{u}_{(D)}}{\lambda_{(D)}} \right) \tag{6}$$

and the data whitening generates new feature matrix $\mathbf{Z} = \mathbf{X}_S\mathbf{W}_2$ in both cases. Data whitening in primal or dual form is preferred in this paper for optional data preprocessing.

2.3 Multiple discriminant analysis

Another approach of dimensionality reduction is based on knowledge of class membership. Having information about classes we can also perform linear data transformation to obtain higher data separation. Classical Fisher discriminant analysis (FDA) (Mika et al. 1999; Croux et al. 2008) is designed for two classes but Rao (Rao and Toutenburg 1995; Duda et al. 2012) generalized it for multi-classification task as follows.

The Rao method transforms the data from $\mathbb{R}^n$ to $\mathbb{R}^{N-1}$ for $N \geq 2$ using a linear transformation

$$\mathbf{z} = \mathrm{RAO}(\mathbf{x}) = \mathbf{W}_3^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_0) \tag{7}$$

where $\mathbf{W}_3 \in \mathbb{R}^{n\times(N-1)}$.

The method is based on pattern index sets $\mathcal{D}_i \in \{k \in \mathbb{N} : y_k^* = i\}$ for $i = 1, ..., N$ and their cardinalities $m_i = \mathrm{card}\,\mathcal{D}_i$. After the evaluation of cluster centres

$$\mathbf{t}_i = \frac{1}{m_i} \sum_{k\in\mathcal{D}_i} \mathbf{x}_k \tag{8}$$

we can calculate within matrix

$$\mathbf{S}_{\mathrm{W}} = \sum_{i=1}^{N} \mathbf{S}_i \geq 0 \tag{9}$$

where

$$\mathbf{S}_i = \sum_{k \in \mathcal{D}_i} (\mathbf{x}_k - \mathbf{t}_i)(\mathbf{x}_k - \mathbf{t}_i)^{\mathrm{T}} \tag{10}$$

The total and between matrices are calculated as

$$\mathbf{S}_{\mathrm{T}} = \sum_{k=1}^{m} (\mathbf{x}_k - \mathbf{x}_0)(\mathbf{x}_k - \mathbf{x}_0)^{\mathrm{T}} \tag{11}$$

$$\mathbf{S}_{\mathrm{B}} = \mathbf{S}_{\mathrm{T}} - \mathbf{S}_{\mathrm{W}} = \sum_{i=1}^{N} m_i (\mathbf{t}_i - \mathbf{x}_0)(\mathbf{t}_i - \mathbf{x}_0)^{\mathrm{T}} \tag{12}$$

When the pattern set is non-degenerated then $\mathbf{S}_{\mathrm{W}} > 0$ and we solve generalized EVD problem which is driven by equation

$$\left(\mathbf{S}_{\mathrm{W}}^{-1}\mathbf{S}_{\mathrm{B}} - \lambda \mathbb{I}_n\right)\mathbf{e} = \mathbf{0}. \tag{13}$$

Solutions of generalized EVD can be ordered as $\lambda_{(1)} \geq \lambda_{(2)} \geq ... \geq \lambda_{(N-1)} \geq 0$ with corresponding eigenvectors $\mathbf{e}_{(1)}, \mathbf{e}_{(2)}, ..., \mathbf{e}_{(N-1)}$. Finally, the transformation matrix of Rao method is

$$\mathbf{W}_3 = (\mathbf{e}_{(1)}, \mathbf{e}_{(2)}, ..., \mathbf{e}_{(N-1)}) \tag{14}$$

and the pattern set has new feature matrix $\mathbf{Z} = \mathbf{X}_{\mathrm{S}}\mathbf{W}_3$ again.

The Rao method is well informed and the new dimension is fixed to $N - 1$ but the data whitening has the optional dimension of result and the information about class membership is missing. Therefore, these two approaches can generate different matrices $\mathbf{W}$ and $\mathbf{Z}$. The Rao method is also useful for data preprocessing.

2.4 DBSCAN technique

There are also various approaches to pattern classification. In this section, we focus on modern sequential clustering algorithms as SLINK, CLINK and finally DBSCAN. The SLINK algorithm (Sibson 1973) carries out single-link cluster analysis on an arbitrary dissimilarity coefficient and provides a representation of the resultant dendrogram which can readily be converted into the usual tree-diagram. There exist also alternative implementation (Goyal et al. 2020) which comes from a reduction in the number of distance calculations required by the standard implementation of SLINK with time complexity O($m \log m$) in the case of m patterns. Hierarchical clustering omitting the initial sorting and consecutive clustering (Schmidt et al. 2017) having a linear time complexity as alternative to single linkage clustering has also been presented.

An algorithm for a complete linkage clustering (Patel et al. 2015) is based, same as SLINK, on a compact representation of a dendrogram. Fast algorithms (Banerjee et al. 2021) for CLINK clustering show that complete linkage clustering of m points can be computed in O($m \log^2 m$) time.

The density-based spatial clustering of applications with noise (DBSCAN) (Ester et al. 1996) represents a non-parametric algorithm with a given set of points in some metric space and groups together points that are closely packed together and marks outliers.

We will study patterns in vector space as $\mathbf{x}_k \in \mathbb{R}^n, k = 1, 2, ..., m$, where m, n are number of patterns and space dimensionality but the DBSCAN is defined in metric space. After

application of Euclidean distance we can define mutual distances as $d_{i,j} = ||\mathbf{x}_i - \mathbf{x}_j||_2$. Various versions of this algorithm (Antony and Deshpande 2016; Bai et al. 2017) differ in the method of distance computation. The inefficient implementations (Shen et al. 2016; Schubert et al. 2017) calculate all mutual distances before data clustering but there are more effective procedures that rapidly decrease the time complexity of DBSCAN to O($m \log m$) as in the case of SLINK.

The DBSCAN is driven by two parameters $\epsilon > 0$, $k_{\min} \geq 2$ which fully depends on users opinion. We will set them to obtain the best sensitivity of resulting classifier in the process of cross-validation. The DBSCAN generates an undirected graph $\mathcal{G}$ with vertex set $\mathcal{V} = \{1, 2, ..., m\}$ and edge set $\mathcal{E} = \{e_1, e_2, ..., e_t\}$ and the pattern $\mathbf{x}_i$ is placed in vertex i for $i = 1, 2, ..., m$. There are three types of vertices: a hard member, a soft member and an outlier.

The vertex i is called the hard member when $\text{card}\{j : d_{i,j} \leq \epsilon\} \geq k_{\min}$. Every edge $e = \{i, j\}$ has to satisfy $d_{i,j} \leq \epsilon$. The edge e is called a hard connection when the vertices i, j are hard members. A soft connection is the edge e where the node i is the hard member but the node j is not. Remaining edges are eliminated. Resulting graph $\mathcal{G}$ has several components. The component is declared as a cluster when it has two vertices at least. Remaining discrete components are declared as the outliers.

The main advantage of SLINK, CLINK, and DBSCAN is in the ability of sequential learning with acceptable time complexity. Exactly, the hard members of DBSCAN, number of clusters, and outliers are invariant to pattern order during the learning process.

2.5 Union of hidden classes

Both supervised and unsupervised approaches to the pattern classification can be used to form hidden classes inside the final classifier. The system of hidden classes arises from uncertainty of class membership, imperfectness of classification or any context out approach. We will apply a deterministic approach which is based on the relationship between hidden groups and the output classes (Hrebik et al. 2019) as follows. The aim is to optimize this relationship as the best mapping from the hidden to the output classes. The strict classifier is defined as mapping $\mathrm{c} : \mathscr{L}_H \to \mathscr{L}_N$ from the set $\mathscr{L}_H$ of hidden class indices to the set $\mathscr{L}_N$ of final class indices, where $\mathscr{L}_n = \{1, ..., n\}$. This mapping can be expressed via the matrix $\mathbb{X} \in \{0, 1\}^{N \times H}$, where $x_{i,j} = 1$ iff $d_k \in \mathscr{H}_j \Rightarrow d_k \in \mathscr{C}_i$. Therefore, $x_{i,j} = 1$ just when for any pattern belonging to $\mathscr{H}_j$ it also belongs to $\mathscr{C}_i$. The uniqueness conditions $\sum_{i=1}^{N} x_{i,j} = 1$ have to be satisfied for $j = 1, ..., H$.

The relation between the classes and the hidden groups is presented via the contingency table $\mathbb{F} \in \mathbb{N}_0^{N \times H}$, where $f_{i,j} = \text{card}\{k : d_k \in \mathscr{C}_i \bigcap \mathscr{H}_j\}$ is the result of pattern counting. Here, $f_{i,j}$ is the number of patterns belonging to both class $\mathscr{C}_i$ and group $\mathscr{H}_j$ as joint frequency, which can be relativized as

$$q_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{H} f_{i,k}} \tag{15}$$

where $i = 1, ..., N$, $j = 1, ..., H$.

There are two approaches to evaluating the quality of a classifier. One of them is accuracy, which is the success of the classifier as a whole, i.e. the proportion of successfully classified patterns and all patterns. The main disadvantage of such approach is the imbalance in results for individual classes. Against this, there is another concept based on

sensitivity, which is very often used in medicine when we evaluate the percentage of success in classifying a sick patient. If, on the other hand, we are interested in the percentage of success in determining whether a patient is healthy, then medicine uses the term specificity. If we do not know in advance how many classes there will be in the task, it is more advantageous to call the classical sensitivity the sensitivity for the first class (se_1) and the specificity the sensitivity of the second class (se_2) and generally introduce se_N, which is actually the success rate of the classifier for the given class, i.e. the number of correct individuals in that class relative to the total number of individuals in that class. It is obvious that if we want to have strict requirements for the classifier, it is not a good idea to maximize its accuracy, but to maximize the so-called critical sensitivity, which is nothing but the smallest value of the individual se_N. When we substitute the values $f_{i,j}$ and $x_{i,j}$ into those definitions, we receive the following formulas.

The accuracy of given classifier can be expressed as

$$acc = \frac{1}{m} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j} \tag{16}$$

Using the concept of class sensitivity as a relative frequency of true classification, we can calculate it for $i = 1, ..., N$ as

$$se_i = \sum_{j=1}^{H} q_{i,j} x_{i,j} \tag{17}$$

An average sensitivity can be defined as

$$ase = \frac{1}{N} \sum_{i=1}^{N} se_i \tag{18}$$

A lower estimate of class sensitivity is defined as a critical sensitivity

$$se^* = \min\{se_i \,:\, i = 1, ..., N\} \tag{19}$$

We prefer critical sensitivity as the strength criterion of classifier efficiency and maximize them via the union of hidden classes. The accuracy criterion is also used as the traditional measure which is frequently used by many authors.

In accordance with Hrebik et al. (2019), we will maximize the critical sensitivity se^*. An adequate mixed binary optimization task is

$$se^* = \max \tag{20}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \text{ for } j = 1, ..., H \tag{21}$$

$$\sum_{j=1}^{H} q_{i,j} x_{i,j} - se^* \geq 0 \text{ for } i = 1, ..., N \tag{22}$$

$$x_{i,j} \in \{0,1\} \text{ for } i = 1,...,N,\ j = 1,...,H \tag{23}$$

$$se^* \in [0,1] \tag{24}$$

with real artificial variable se^*. The inequalities Eq. (22) guarantee that se^* is a lower bound of critical sensitivity during the optimization process.

From the theoretical point of view it is necessary to think whether this task has a solution in general. If we choose $se^* = 0$ then all inequalities Eq. (22) holds regardless of what the values of x are. This means that if the task was too complicated, then in the worst possible scenario it will case that the optimal value of critical sensitivity will be $se^* = 0$, as a symbol that the original task has no solution, but the system of inequalities Eq. (22) has a solution. So, if we obtain $se^* = 0$, it means that the given task cannot be solved by the given method. In the case of degeneration, the task can have more solution. Therefore is important following consideration.

After the specification of se^*, we can yield from the task degeneration and solve additional binary programming task which guarantees the same critical sensitivity and maximize accuracy as

$$acc = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j} = \max \tag{25}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \text{ for } j = 1,...,H \tag{26}$$

$$\sum_{j=1}^{H} q_{i,j} x_{i,j} \geq se^* \text{ for } i = 1,...,N \tag{27}$$

$$x_{i,j} \in \{0,1\} \text{ for } i = 1,...,N,\ j = 1,...,H \tag{28}$$

3 Framework of multi-classification

The novel approach of vector pattern multi-classification is based on a combination of the approaches mentioned above. Basic assumptions and procedures are summarized in this section.

3.1 Assumptions

Let N, M, n be number of output classes, number of patterns, and number of pattern dimensions that are unlimited in general. But there is a threshold value n^* of pattern dimension which switch between deterministic and random sub-sampling approaches. In both cases, the first step of classification is dimensionality reduction using data whitening or multi-class discriminant analysis which converts the data into the space of dimension $D \leq n$. In the second step, the reduced data are clustered using the DBSCAN technique and the

hidden classes are formed. Optimal unions of these hidden classes are performed in the last step of the multiple classifications.

3.2 Classification for $n \leq n^*$

The learning strategy of classification is based on the pattern dimension. When the patterns are not too large we will proceed with the whole set of patterns. In the case of data whitening we use learning procedure with parameters $D, k_{\min}, \epsilon$. But in the case of multi-class discriminant analysis, we have only two free parameters $k_{\min}, \epsilon$ because of $D = N - 1$. In the first step we transform data matrix $\mathbf{X} \in \mathbb{R}^{M \times n}$ to $\mathbf{Y} \in \mathbb{R}^{M \times D}$ using data whitening Eqs. (1–5) or using Rao method Eqs. (7–14). Then we use the DBSCAN technique for clustering in $\mathbb{R}^D$ with parameters $k_{\min}, \epsilon$. Every cluster forms a new hidden class of patterns and the outliers which are also localized by the DBSCAN are ignored. Finally, the optimal union of hidden classes Eqs. 20–24) is performed.

There are no problems with large pattern number M because both data whitening, Rao method, and DBSCAN are designed for a large amount of data patterns. But the parameters of DBSCAN must be selected to generate not too large number H of hidden classes.

3.3 Approximated classification for $n > n^*$

When the pattern vector length n is too large its reduction is necessary preprocessing step which is a kind of context out data whitening. We select $n_{\text{red}} \leq n^*$ first and create a random sub-sample of m patterns which are supposed to be representatives of the given pattern set. When $m \leq n^*$, the dual form of learning Eq. (6) is performed as an alternative data whitening which produces the weight matrix $\mathbf{W} \in \mathbb{R}^{n \times n_{\text{red}}}$. Using this matrix we transform the original data to obtain a matrix $\mathbf{X}_{\text{red}} \in \mathbb{R}^{M \times n_{\text{red}}}$. This matrix of reduced patterns is used instead of the original pattern set using the learning strategy (Sect. 3.2). This process is of a stochastic nature and n_{red}, m are two additive parameters that control the preliminary dimensionality reduction. Therefore, the novel classification algorithm is also applicable to long pattern vectors but with context out imperfectness related to the random sampling of patterns.

3.4 Classification verification

We suppose the parameters of classification are set on the complete pattern set to obtain the classifier with maximum possible critical sensitivity se^*. The role of parameter ϵ for fixed $D, k_{\min}$ is crucial and can rapidly change the class sensitivities but the critical sensitivity peaceful-wise continuous function of ϵ and therefore there is an interval of ϵ which maximizes se^*. After this preliminary parameter setting we have to perform the cross-validation. When the number of patterns M is small, we prefer Leave-One-Out (Wong 2015; Gronau and Wagenmakers 2019) cross-validation technique but for large M we can use 10-fold (Xu et al. 2018; Steyerberg 2019) cross-validation scheme as generally recommended. When we map the role of parameter ϵ in the case of cross-validation the values of se^* are not so high in many cases. The interval of an optimal ϵ can be also different in the case of cross-validation. There are no general rules and this phenomenon will be studied experimentally in the next section.

3.5 Classifier structure

The new classifier serves to ensure that for all patterns $\mathbf{x}_k \in \mathbb{R}^n, k = 1, 2, ..., m$, where m, n are number of patterns and space dimensionality, recognized to which class $\mathscr{C}_i, i = 1, 2, ..., N$ belongs. The new classifier consists of three parts, and the proposed structure is captured in Fig. 1. The first part of the system, is a linear transformation that displays all patterns from dimension n to a lower dimension D. Either multiple discriminant analysis (Sect. 2.3) or data whitening (Sect. 2.2) is used for this transformation. The second part of the system uses the standard DBSCAN tool with parameters ϵ and k_{min} (Sect. 2.4). Depending on these parameters, individual patterns are classified unsupervised into H classes, representing hidden classes. The third part of the system is optimal union using binary programming techniques (Sect. 2.5). This creates a system that unambiguously assigns each x on the input to which class it belongs.

4 Experimental part

To demonstrate the results of proposed approach we have selected ten basic classification tasks (Dua 2020). This approach allow clear comparison with other classification methods. All datasets analysed during the current study are available in the repositories referred in Table 1, mainly in UC Irvine Machine Learning Repository (Dua 2020). The table includes also the number of patterns, their dimensionality, and the number of classes. For comparison of our results, we work with relevant papers presenting classification results. The main aim is to confirm that our method is comparable with other ones. As our approach based on critical sensitivity is not commonly used we compare the basic criteria of classification accuracy. We present and compare both cases' optimal setting on the training set as well as the leave-one-out cross-validation.

4.1 Case study: iris flower classification task

As the primal research dataset, we decided to use the well-known and widely used iris dataset (Swain et al. 2012). Iris dataset contains three classes of fifty instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two, the latter is not linearly separable from each other. Every iris pattern is a four-dimensional real positive vector.

First of all, we have to prepare the data for our model. We can use the data whitening or Rao method. Our aim is to demonstrate using of DBSCAN. We test different values of ϵ

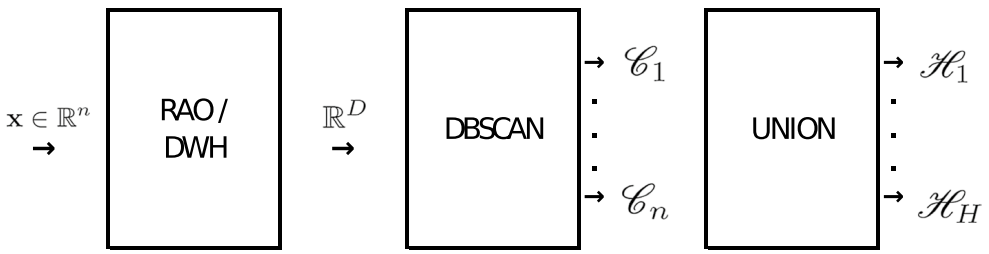


Fig. 1 Classifier structure

Table 1 Datasets for Assessment

Dataset	m	n	H
Iris (Dua 2020)	150	4	3
Wine (Dua 2020)	178	13	3
Glass (Dua 2020)	214	9	2
Cancer (Wisconsin) (Dua 2020; Basavegowda and Dagnew 2020)	699	9	2
Haberman (Dua 2020)	306	3	2
Liver (Dua 2020)	345	6	2
Ionosphere (Dua 2020)	351	33	2
Cancer (Coimbra) (Dua 2020; Patrício et al. 2018)	116	10	2
Transfusion (Dua 2020; Yeh et al. 2009)	748	4	2
Cryotherapy (Dua 2020; Khozeimeh et al. 2017, 2017)	90	6	2

and investigate results for $k_{\min}$ from 2 to 5. The ϵ values in the testing examples we set from 0 to 1.26. The optimal settings with the highest value of se^* are summarized in Table 2 for training set and in Table 3 for Leave-One-Out cross-validation.

We can confirm the best results using $k_{\min} = 2$, Rao method and ϵ from $\epsilon_{\min} = 0.08$ to $\epsilon_{\max} = 0.09$ in naive case. Leave-One-Out cross-validation led to the wider interval of ϵ reaching 0.04 to 0.14 for the same setting. The value of critical sensitivity is above ninety percent in both cases. The reached maximum accuracy value is fully comparable to other presented results (Kulluk et al. 2012; Ozyildirim and Avci 2014; Asafuddoula et al. 2017; Yin and Gelenbe 2018) summarized in Tables 4 and 5 along with reached minimum and maximal accuracy together with placement among benchmark techniques.

4.2 Application to other pattern sets

We have applied our novel method also to nine other datasets: Wine, Glass, Cancer (Wisconsin), Haberman, Liver, Ionosphere, Cancer (Coimbra), Transfusion, and Cryotherapy described in Table 1. Most of the tasks are classification into two classes. In the case of the Glass dataset, there is known also an alternative task to seven classes. The results of training are collected in Table 2 as optimal values of classification parameters and adequate

Table 2 Optimal setting on training set

Dataset	D	$k_{\min}$	$\epsilon_{\min}$	$\epsilon_{\max}$	acc	ase	$se*$
Iris	0	2	0.0832	0.0912	0.9800	0.9800	0.9600
Wine	3	2	0.2188	0.3468	0.7809	0.7820	0.7746
Glass	2	2	0.2042	0.3981	0.9626	0.9687	0.9571
Cancer (Wisconsin)	1	2	0.0126	0.0145	0.9757	0.9765	0.9738
Haberman	0	2	0.0130	0.0440	0.7582	0.7605	0.7556
Liver	0	2	0.0224	0.0501	0.7536	0.7582	0.7517
Ionosphere	0	2	0.0016	0.0029	0.9203	0.9203	0.9200
Cancer (Coimbra)	0	2	0.0316	0.1148	0.7931	0.7945	0.7813
Transfusion	0	2	0.0028	0.0759	0.7580	0.7561	0.7640
Cryotherapy	2	2	0.0050	0.0912	0.9556	0.9554	0.9524

Table 3 Optimal setting under leave-one-out cross-validation

Dataset	D	$k_{\min}$	$\epsilon_{\min}$	$\epsilon_{\max}$	acc	ase	$se*$
Iris	0	2	0.0426	0.1445	0.9533	0.9533	0.9200
Wine	3	2	0.2188	0.2454	0.7416	0.7437	0.7292
Glass	2	2	0.2512	0.3802	0.9439	0.9295	0.9019
Cancer (Wisconsin)	1	2	0.0100	0.0501	0.9700	0.9692	0.9668
Haberman	0	3	0.0010	0.0021	0.7124	0.5829	0.3704
Liver	0	2	0.0018	0.0631	0.6841	0.6626	0.6552
Ionosphere	0	2	0.0135	0.234	0.8832	0.8530	0.7460
Cancer (Coimbra)	0	2	0.0851	0.1318	0.7069	0.7056	0.6923
Transfusion	0	2	0.0010	0.0097	0.6865	0.6279	0.5506
Cryotherapy	2	2	0.0224	0.0912	0.9556	0.9554	0.9524

values of accuracy and sensitivities. The proposed method has been also treated by leave-one-out cross-validation. Obtained results are summarized in Table 3 in a similar way. There are no dramatic changes in parameter setting and resulting accuracy and sensitivities in Tables 2 and 3. The SLINK algorithm as DBSCAN for $k_{\min} = 2$ is a good choice in a majority of cases. But the type dimension of primal coordinate reduction is task sensitive. The Rao technique multi-class discriminant analysis is useful in many cases.

As seen in Tables 4 and 5 our method is comparable with standard techniques of classification. To see how competitive our results are, we went through several papers on this topic (Kulluk et al. 2012; Ozyildirim and Avci 2014; Rani and Ganesh 2014; Abdar et al. 2017; Asafuddoula et al. 2017; Kahramanli 2017; Aslan et al. 2018; Li and Chen 2018; Talabni and Engin 2018; Yin and Gelenbe 2018; Austria et al. 2019; Chan and Chin 2019; Kraipeerapun and Amornsamankul 2019; Rahman et al. 2020). The rank of our novel method has been evaluated for every dataset and during both training and cross-validation processes. Comparison results including the rank among others is in Tables 4 and 5. The proposed method is in the second quartile related to the involved referential methods in most cases. As the use of datasets varies, also the number of used benchmark techniques for comparison is variant and summarized in the following paragraphs.

Table 4 Accuracy compared classifiers for training

Dataset	Training			
	Proposed	Best	Worse	Rank
Iris	0.9800	0.9820	0.7993	2/18
Wine	0.7809	1.0000	0.3952	13/18
Glass	0.9626	0.9930	0.3546	2/11
Cancer (Wisconsin)	0.9757	1.0000	0.8920	5/14
Haberman	0.7582	0.7850	0.7353	4/9
Liver	0.7536	0.9233	0.5043	7/16
Ionosphere	0.9263	1.0000	0.8664	7/11
Cancer (Coimbra)	0.7931	0.8000	0.7350	3/5
Transfusion	0.7580	0.9318	0.7500	3/5
Cryotherapy	0.9556	0.9590	0.8540	2/11

Table 5 Accuracy compared classifiers for cross-validation

Dataset	Testing (cross-validation)			
	Proposed	Best	Worse	Rank
Iris	0.9533	1.0000	0.8200	8/18
Wine	0.7416	0.9882	0.3539	11/17
Glass	0.9439	0.9660	0.3084	4/13
Cancer (Wisconsin)	0.9700	0.9700	0.7077	1/22
Haberman	0.7124	0.7843	0.6111	5/10
Liver	0.6841	0.9883	0.4174	12/19
Ionosphere	0.8832	0.9375	0.5755	13/20
Cancer (Coimbra)	0.7069	0.7430	0.5814	7/16
Transfusion	0.6865	0.7861	0.5849	6/8
Cryotherapy	0.9556	0.9778	0.5372	4/12

According to Kulluk et al. (2012) for training and cross-validation comparison we used five different variants of harmony search algorithms and standard backpropagation algorithm. For training, we compared the results with approximator using a spiking random neural network using five other benchmarks (Yin and Gelenbe 2018). For cross-validation results comparison we used generalized classifier neural network and its logarithmic learning implementation, probabilistic neural network and standard multilayer perceptron as presented in Ozyildirim and Avci (2014). The results of an incremental ensemble classifier method (Asafuddoula et al. 2017) were also used as a benchmark.

Comparison of training and testing for Haberman and Liver datasets is included for product-unit neural networks as a special class of feed-forward neural network, together with results for backpropagation and Levenberg–Marquardt algorithms, as presented in Kahramanli (2017).

For Ionosphere and Transfusion datasets, we used also method proposed in Chan and Chin (2019) to tackle the problem of imbalanced data based on cosine similarity together with results for synthetic minority oversampling technique and adaptive synthetic sampling approach. The Liver dataset is compared to the results of two proposed methods Boosted C5.0 and CHAID based on the decision trees and five other methods (backpropagation, NB tree, decision tree, C5.0, support vector machine, and basic neural network) presented in Abdar et al. (2017). The training results for Transfusion dataset were obtained by the naive Bayesian classifier, implementation of algorithm iterative Dichotomiser 3, and random tree, all presented in Rani and Ganesh (2014).

Both breast cancer datasets, Cancer Wisconsin and Coimbra, were compared with five different classification models including decision tree, random forest, support vector machine, neural network and logistics regression as presented in Li and Chen (2018). Coimbra dataset is compared with results of an artificial neural network, standard extreme learning machine, support vector machine and K-nearest neighbour presented in Aslan et al. (2018), and additionally to ten classification algorithms and their variations including logistic regression, k-nearest neighbour, support vector machine, decision tree, random forest, gradient boosting method, and naive Bayes presented in Austria et al. (2019).

In the case of Cryotherapy dataset, we used comparison with four methods using kernel functions for improving the learning capacity of support vector machine presented in Talabni and Engin (2018). Seven methods additional methods for Cryotherapy, two methods

based on the combination of cascade generalization and complementary neural network, and five existing methods (neural network, stacked generalization, cascade generalization, complementary neural network, the combination of stacked generalization as presented in Kraipeerapun and Amornsamankul (2019). Cryotherapy results were also compared with proposed using of support vector machine and nine standard methods k-nearest neighbours, binary logistic regression, linear discriminant analysis, quadratic discriminant analysis, classification and regression trees, random forest, adaptive boosting, gradient boosting, and bagging, presented and summarized in Rahman et al. (2020).

Results for training using Iris, Wine, Glass, Haberman, Liver and Ionosphere datasets are compared also with 1-NN classifier and two its variants, namely the Hypersphere Classifier and the Adaptive Nearest Neighbor Rule, as presented in Orozco-Alzate et al. (2019). Results of Fuzzy Pattern Trees using Grammatical Evolution, called Fuzzy Grammatical Evolution (Murphy et al. 2022), is used for comparison on Iris, Wine, Haberman and Transfusion datasets.

For additional comparison we have included also average accuracy results for cross-validation of scalable ensemble technique XGBoost, random forest, gradient boosting, LightGBM using selective sampling of high gradient instances and Ordered CatBoost modifiing the computation of gradients to avoid the prediction shift as presented in Bentejac et al. (2021) for Iris, Wine, Cancer (Wisconsin), Liver, and, Ionosphere. Cross-validation results in case of hybrid classification model named HyCASTLE (Veneri et al. 2022) is also included for Iris, Glass and Haberman.

Based on all previous experiments we see our method as quite robust. Robustness means a similar parameter setting in the training and cross-validation processes in this case. The novel method is therefore comparable with current methods and its learning is very simple and robust related to the parameter value. Moreover, we present additionally the values of critical sensitivities, which we believe are relevant for the classifier quality.

5 Conclusion

The proposed type of pattern classifier with embedded dimensionality reduction and hidden class forming has been tested. Based on training and cross-validation using ten standard datasets, the new type of classifier has several advantages evaluated below.

Using a training materialized for direct verification or leave-one-out cross-validation, we set DBSCAN parameter $k_{\min} = 2$ in most cases to obtain the best value of critical sensitivity of a given system. Therefore, there is no need to use general DBSCAN approach because its reduced version the clustering algorithm SLINK produces less compact hidden clusters. It is usually the main weakness of the SLINK approach, but in our case, the optimal unioning of hidden classes eliminates this disadvantage. The future realization of this classifier can use only SLINK instead.

The most important property of this classifier is similar behavior during learning on the training set and the cross-validation. We suppose that selected method of dimensionality reduction and length of resulting vectors depend only on the dataset but are independent of the verification strategy.

There is also a similarity in the range of parameter $\epsilon[\epsilon_{\min}, \epsilon_{\max}]$. Moreover, the optimal ϵ is included in the optimum range of cross-validation. Therefore the optimal set of parameter ϵ can be directly used as relevant estimate of ϵ for the leave-one-out cross-validation processes.

The proposed method aims the critical sensitivity maximization, which produces non-trivial unions of hidden classes. But the authors of referential techniques are oriented mainly to the accuracy evaluation of classifiction, which complicates the method comparison.

There are several general recommendations for the setting of a classifier. The user decides first whether to apply multi-class discriminant analysis or whether he prefers data whitening of given dimension D. The SLINK method with parameter ϵ is suggested for hidden class forming. The optimal values of D and ϵ can be estimated using only the training set for verification. Finally, the leave-one-out cross-validation is a simple process focused only on parameter ϵ improvement, which remains unchanged in many cases. The proposed classifier, represents a comparable alternative to other pattern classification techniques focusing on critical sensitivity.

The main advantage of the proposed classifier is that it successfully avoids the curse of dimensionality and includes automatic reduction and standardization of the input dataset. All cluster analysis is carried out in dimensionless coordinates and thus offers a wide range of uses for a whole range of applications. It is a relatively simple classifier having comparable properties comapred to more complicated ones. As one of the other applications, we can recommend, for example, the recognition of unstructured data such as strings, trees, and graphs. In such a case, it is necessary to set correctly a suitable feature description, which precedes the reduction layer of our classifier.

Acknowledgements The authors would like to acknowledge the support of the research grant SGS20/190/OHK4/3T/14. The second author also acknowledges the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 Research Center for Informatics.

Funding Open access publishing supported by the National Technical Library in Prague.

Data availability The datasets generated during and/or analysed during the current study are available in the UC Irvine Machine Learning Repository (Dua 2020).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Abdar M, Zomorodi-Moghadam M, Das R, Ting IH (2017) Performance analysis of classification algorithms on early detection of liver disease. Exp Syst Appl 67:239–251

Antony N, Deshpande A (2016) Domain-driven density based clustering algorithm. Proceedings of international conference on ICT for sustainable development. Springer, pp 705–714

Asafuddoula M, Verma B, Zhang M (2017) An incremental ensemble classifier learning by means of a rule-based accuracy and diversity comparison. International joint conference on neural networks. IEEE, pp 1924–1931

Aslan MF, Celik Y, Sabanci K, Durdu A (2018) Breast cancer diagnosis by different machine learning methods using blood analysis data. Int J Intell Syst Appl Eng 6(4):289–293
Austria YD, Lalata JAP, Maria LB Jr, Goh JEE, Goh MLI, Vicente HN (2019) Comparison of machine learning algorithms in breast cancer prediction using the coimbra dataset. Int J Simul Syst Sci Technol 20:23
Back T, Fogel DB, Michalewicz Z (2018) Evolutionary computation 1: basic algorithms and operators. CRC Press
Bai L, Cheng X, Liang J, Shen H, Guo Y (2017) Fast density clustering strategies based on the k-means algorithm. Pattern Recognit 71:375–386
Banerjee P, Chakrabarti A, Ballabh TK (2021) An efficient algorithm for complete linkage clustering with a merging threshold. Data management, analytics and innovation. Springer, pp 163–178
Basavegowda HS, Dagnew G (2020) Deep learning approach for microarray cancer data classification. CAAI Trans Intell Technol 5(1):22–33
Bentejac C, Csorgo A, Martinez-Munoz G (2021) A comparative analysis of gradient boosting algorithms. Artif Intell Rev 54(3):1937–1967
Chan TK, Chin CS (2019) Health stages diagnostics of underwater thruster using sound features with imbalanced dataset. Neural Comput Appl 31(10):5767–5782
Croux C, Filzmoser P, Joossens K (2008) Classification efficiencies for robust linear discriminant analysis. Stat Sin 2008:581–599
Dua D, Graff C (2020) UCI machine learning repository (2020). http://archive.ics.uci.edu/ml
Duda RO, Hart PE, Stork DG (2012) Pattern classification. Wiley
Eldar YC, Oppenheim AV (2003) Mmse whitening and subspace whitening. IEEE Trans Info Theory 49(7):1846–1851
Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd 96:226–231
Goyal P, Kumari S, Sharma S, Balasubramaniam S, Goyal N (2020) Parallel slink for big data. Int J Data Sci Anal 9(3):339–359
Gronau QF, Wagenmakers E-J (2019) Limitations of bayesian leave-one-out cross-validation for model selection. Comput Brain Behav 2(1):1–11
Hrebik R, Kukal J, Jablonsky J (2019) Optimal unions of hidden classes. Cent Euro J Op Res 27(1):161–177
Hu G, Yin C, Wan M, Zhang Y, Fang Y (2020) Recognition of diseased pinus trees in uav images using deep learning and adaboost classifier. Biosyst Eng 194:138–151
Jafarzadeh H, Mahdianpari M, Gill E, Mohammadimanesh F, Homayouni S (2021) Bagging and boosting ensemble classifiers for classification of multispectral, hyperspectral and polsar data: a comparative evaluation. Remote Sens 13(21):4405
Jolliffe I (2011) Principal component analysis. Springer
Kahramanli H (2017) Training product-unit neural networks with cuckoo optimization algorithm for classification. Int J Intell Syst Appl Eng 5(4):252–255
Karlsson C (2010) Handbook of research on cluster theory. Edward Elgar Publishing
Khozeimeh F, Alizadehsani R, Roshanzamir M, Khosravi A, Layegh P, Nahavandi S (2017) An expert system for selecting wart treatment method. Comput Biol Med 81:167–175
Khozeimeh F, Jabbari Azad F, Mahboubi Oskouei Y, Jafari M, Tehranian S, Alizadehsani R, Layegh P (2017) Intralesional immunotherapy compared to cryotherapy in the treatment of warts. Int J Dermatol 56(4):474–478
Kraipeerapun P, Amornsamankul S (2019) Using cascade generalization and neural networks to select cryotherapy method for warts. 2019 International conference on engineering, science, and industrial applications (ICESI). IEEE, pp 1–5
Kulluk S, Ozbakir L, Baykasoglu A (2012) Training neural networks with harmony search algorithms for classification problems. Eng Appl Artif Intell 25(1):11–19
Li Y, Chen Z (2018) Performance evaluation of machine learning methods for breast cancer prediction. Appl. Comput. Math 7(4):212–216
Liang T, Sur P (2022) A precise high-dimensional asymptotic theory for boosting and minimum-l1-norm interpolated classifiers. Ann Stat 50(3):1669–1695
Lin H, Zhao B, Liu D, Alippi C (2020) Data-based fault tolerant control for affine nonlinear systems through particle swarm optimized neural networks. CAA J Autom Sin 7(4):954–964
Liu F, Wang J (2022) An accurate method of determining attribute weights in distance-based classification algorithms. Math Probl Eng. https://doi.org/10.1155/2022/6936335
Medina-Pérez MA, Monroy R, Camiña JB, García-Borroto M (2017) Bagging-tpminer: a classifier ensemble for masquerader detection based on typical objects. Soft Comput 21(3):557–569

Mika S, Ratsch G, Weston J, Scholkopf B, Mullers KR (1999) Fisher discriminant analysis with kernels. Neural networks for signal processing. IEEE, pp 41–48

Murphy A, Ali MS, Mota Dias D, Amaral J, Naredo E, Ryan C (2022) Fuzzy pattern tree evolution using grammatical evolution. SN Comput Sci 3(6):1–13

Nguyen LH, Holmes S (2019) Ten quick tips for effective dimensionality reduction. PLoS Comput Biol 15(6):1006907

Orozco-Alzate M, Baldo S, Bicego M (2019) Relation, transition and comparison between the adaptive nearest neighbor rule and the hypersphere classifier. International conference on image analysis and processing. Springer, pp 141–151

Ozyildirim BM, Avci M (2014) Logarithmic learning for generalized classifier neural network. Neural Netw 60:133–140

Patel S, Sihmar S, Jatain A (2015) A study of hierarchical clustering algorithms. 2nd International conference on computing for sustainable global development. IEEE, pp 537–541

Patrício M, Pereira J, Crisóstomo J, Matafome P, Gomes M, Seicca R, Caramelo F (2018) Using resistin, glucose, age and bmi to predict the presence of breast cancer. BMC Cancer 18(1):29

Rahman M, Zhou Y, Wang S, Rogers J et al (2020) Wart treatment decision support using support vector machine. University of Texas

Rani SA, Ganesh SH (2014) A comparative study of classification algorithm on blood transfusion. J Adv Res Technol 3:57–60

Rao CR, Toutenburg H (1995) Linear models. Springer, pp 3–18

Rekha G, Madhu S (2022) An hybrid approach based on clustering and synthetic sample generation for imbalance data classification: clustsyn. Proceedings of data analytics and management. Springer, pp 775–784

Schmidt M, Kutzner A, Heese K (2017) A novel specialized single-linkage clustering algorithm for taxonomically ordered data. J Theor Biol 427:1–7

Schubert E, Sander J, Ester M, Kriegel HP, Xu X (2017) Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Trans Database Syst (TODS) 42(3):19

Shahid AH, Singh M (2019) Computational intelligence techniques for medical diagnosis and prognosis: problems and current developments. Biocybern Biomed Eng 39(3):638–672

Shen J, Hao X, Liang Z, Liu Y, Wang W, Shao L (2016) Real-time superpixel segmentation by dbscan clustering algorithm. IEEE Trans Image Processing 25(12):5933–5942

Shi G, Zhao B, Li C, Wei Q, Liu D (2019) An echo state network based approach to room classification of office buildings. Neurocomputing 333:319–328

Sibson R (1973) Slink: an optimally efficient algorithm for the single-link cluster method. Comput J 16(1):30–34

Steyerberg EW (2019) Validation of prediction models. Springer, pp 329–344

Swain M, Dash SK, Dash S, Mohapatra A (2012) An approach for iris plant classification using neural network. Int J Soft Comput 3(1):79

Talabni H, Engin A (2018) Impact of various kernels on support vector machine classification performance for treating wart disease. International conference on artificial intelligence and data processing. IEEE, pp 1–6

Veneri MD, Cavuoti S, Abbruzzese R, Brescia M, Sperlì G, Moscato V, Longo G (2022) Hycastle: A hybrid classification system based on typicality, labels and entropy. Knowl Based Syst 244:108566

Wong TT (2015) Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. Pattern Recognit 48(9):2839–2846

Xu L, Fu HY, Goodarzi M, Cai CB, Yin QB, Wu Y, Tang BC, She YB (2018) Stochastic cross validation. Chemom Intell Lab Syst 175:74–81

Yeh IC, Yang KJ, Ting TM (2009) Knowledge discovery on rfm model using bernoulli sequence. Exp Syst Appl 36(3):5866–5871

Yin Y, Gelenbe E (2018) A classifier based on spiking random neural network function approximator. Preprint available in ResearchGate. net

Neural Processing Letters (2022) 54:835–852
https://doi.org/10.1007/s11063-021-10660-1

Diffusion Modelling

Topographic Error of SOM Under Control

Radek Hrebik[1] · **Jaromir Kukal**[1]

Accepted: 6 October 2021 / Published online: 1 November 2021

Abstract
The traditional self-organized map (SOM) is learned by Kohonen learning and the most common 2-dimensional grids defining the structure of the map are the hexagonal grid and the rectangular grid. A novel model of self-organization is based on hexagonal grid and diffusion modeling in continuous space which is a good approximation of endorphins propagation and nitric oxide generation in the real brain. Therefore the structure of the system is described by neuron coordinates instead of neighborhood relationships in traditional SOM. The discussed neuron activation using the diffusion process and novel diffusive learning algorithm is based on this activation mentioned above. The novel structure and algorithm are demonstrated on simple examples and real economic applications.

Keywords Self organization · Kohonen map · Diffusion learning · SOM · Topographic error

1 Introduction

There are many approaches to how to perform modeling of self-organization. We analyze self-organization with a given grid, and therefore we focus on the self-organized maps (SOM). The models can be directly inspired by the anatomy and physiology of the neuronal system or rather by other ideas that are easy to realize. Our research is inspired by the Pudding model of an atom in physics [3,33], where the nucleus of atoms are supposed as points (raisins) in the electron continuum (pudding). Using self-organization, we will place individual neurons instead of the atom nucleus into the continuum that would transfer the information in the system. The second inspiration is strongly related to brain physiology study concerning slow signal propagation in the central nervous system.

Self-organized maps are useful in digital image processing and image denoising as well [34]. The weights of the neurons in a trained SOM can also help when choosing whether to shrink or expand the current contour during the optimization process performed itera-

✉ Radek Hrebik
Radek.Hrebik@seznam.cz

Jaromir Kukal
Jaromir.Kukal@fjfi.cvut.cz

[1] FNSPE, CTU in Prague, Brehova 7, 115 19 Prague 1, Czech Republic

tively. The model can handle images that contain objects characterized by complex intensity distributions [1].

Nitric oxide (NO) represents the final product of endorphins degradation and a well-known neurotransmitter in the mammal brain due to its ability to diffuse isotropically in aqueous and lipid environments [28]. Using NO as an intracellular signaling molecule in the nervous system has been confirmed by many studies [13,18]. The information transmission by neurons, in vertebrates and invertebrates, has also been discussed by many authors [9,12, 14,26]. The ways of intervening cellular or membrane structures discussed in [22,35]. The whole surface of the neuron is, therefore, a potential release site for NO. In marked contrast to conventional transmitter release, being restricted to the synaptic zone [17,27,32].

These physiological studies serve as a background model for the realization of artificial self-organization systems. Lopez et al. [23,24] developed two pure informatics models yielding from the simplification of nitric oxide dynamic but not focused on the physical description of the diffusion process. Moreover, in these studies, the spatial effect is modeled as a multi-compartment discrete system.

Previous studies motivate us to introduce a novel model of SOM learning. In contrast to them, we focus on unbounded space diffusion with a simple chemical reaction. The main advantages of the novel model are:

- The SOM neurons being in any Euclidean space, moreover, irregular spacing is also permitted.
- The diffusion process enables modeling short and long-distance learning without the necessity of the SOM topology declaration.
- The nitric oxide is also slowly degraded by following chemical reaction avoiding its cumulation in the brain.
- Model linearity and absence of finite boundaries when spreading the chemical reaction guarantee the existence of a fundamental solution of model equations in a simple form, which is easy to pre-calculate to save the time complexity of a novel learning algorithm.

Our motivation was to design SOM with free but fixed positions of individual neurons. In this case, the neuron distances are not integer as in traditional graph SOM are, which enables a more sensitive perception of SOM distances as Euclidean once. The cardinal question is how to design the SOM model with this property but in the simple possible form. The main advantage of the novel method is in the model simplicity and generality. But the novel SOM model is also a simplification of the slow information transfer mechanism in the brain. Individual neurons are supposed to be generators of nitric oxide propagated by the diffusion process with a chemical reaction of the first order as the simplest model of nitric oxide degradation. The simplification of biological reality is in several assumptions. The brain matter is supposed to be a continuum in the physical sense, and the space is unconstrained. The other chemical substances like endorphins and intermediates are organic molecules of larger size, and their diffusion is extraordinarily slow. Therefore, the only diffusion of small nitric oxide molecules is supposed in this simple model of the brain. The biological background of a new model is crucial for result interpretation driven by natural processes.

In this paper, we came back to the origins of diffusion [5,6] and primal neurophysiological studies [13,18] to obtain a novel structure and adequate learning algorithm of pudding SOM, as a simplification of the real nitric oxide diffusion process. The proposed novel SOM method, based on a simple diffusion model, will be described in the next section.

2 Diffusion with Chemical Reaction

The slow information transfer in the nervous system can be modeled as a diffusion process [6] with first-order chemical reaction [10]. The reactant generated by single-neuron activity and the diffusion process spread the substance [3] in the neuron neighborhood. Our model derived from the classical diffusion process in the unconstrained continuum built on a traditional substance behavior model, described by the second Fick's law [4]. Due to the degradation of nitric oxide, the modified diffusion model by the kinetics of pseudo-monomolecular [31] chemical reaction. At the beginning of the learning process, zero concentration of nitric oxide is supposed. The neuron activity causes endorphin generation and, therefore also nitric oxide appears close to the given neuron. From the concentration point of view, it is only a Dirac impulse at a given point and time. These simplifications enable us to obtain an analytical solution of the diffusion equation and following a dimensionless approach which will reduce the number of model parameters. But these advantages will be lost when the reaction kinetic is nonlinear, and the diffusion coefficient is a function of concentration, or space is geometrically constrained. We can obtain the analytical solution as follows.

Let $N \in \mathbb{N}$ be space dimension, $\mathbf{y} \in \mathbb{R}^N$ be point coordinate, $D, t, \lambda > 0$ be diffusion coefficient, time and rate constant of chemical reaction. The free diffusion of reacting substrate of concentration $\mathrm{c} : \mathbb{R}^N \to \mathbb{R}_0^+$ is driven by partial differential equation

$$\frac{\partial \mathrm{c}(\mathbf{y}, t)}{\partial t} = D \bigtriangledown^2 \mathrm{c}(\mathbf{y}, t) - \lambda \mathrm{c}(\mathbf{y}, t) \tag{1}$$

with initial condition

$$\mathrm{c}(\mathbf{y}, 0_+) = \delta(\mathbf{y}), \tag{2}$$

where δ is Dirac function. The free diffusion is constraint by boundary condition

$$\lim_{\|\mathbf{y}\| \to +\infty} c(\mathbf{y}, t) = 0 \tag{3}$$

The fundamental solution of (1) is

$$\mathrm{c}(\mathbf{y}, t) = \frac{1}{(4\pi Dt)^{N/2}} \cdot \exp\left(-\frac{\|\mathbf{y}\|_2^2}{4Dt}\right) \cdot \exp(-\lambda t). \tag{4}$$

Due to the system linearity, time, and space invariance of (1), we can use the fundamental solution to the study of multi-neuron systems with sequential activities. We can describe the novel model in a more formal style based on previous formulas of physical and chemical origin.

3 Pudding Model of SOM

Our model of the self-organized map based on specific assumptions:

- Finite number of neurons placed in fixed positions like raisins in a pudding.
- Unconstrained continuum surrounding the neurons as an analogy of the pudding base.
- Neuron interconnections omitted.
- The pattern set stays outside the pudding and only sequentially activates individual neurons.
- Neuron activities generate the concentration profile of the substrate in the pudding.

– The nitric oxide as neurotransmitter takes part in complex biochemical kinetics of axon recovery.
– Substrate concentration influences the learning rates of individual neurons.
– The learning process changes weights as neuron properties.

The *Pudding model* description begins with remembering basic facts. Let $m, n, H \in \mathbb{N}$ be the number of patterns, pattern dimensionality, and the number of SOM neurons [15]. The individual patterns are $\mathbf{x}_j \in \mathbb{R}^n$, where $j = 1, \ldots, m$ and form the pattern set $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. The fixed positions of individual neurons in continuum are $\mathbf{p}_i \in \mathbb{R}^N$ for $i = 1, \ldots, H$ and reflects the topology of SOM [25] which is subject of network design. The diffusion process in continuum can be easily expressed using matrix $\mathbf{D} \in (\mathbb{R}_0^+)^{H \times H}$ of distances $d_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\|_2$. These mutual distances indirectly express the topology of SOM. The traditional Kohonen SOM is based on a connectivity graph, where the neuron distances are integers as the traditional distances in an undirected graph. The Pudding SOM is comparable with traditional SOM only when the nearest neighbor neurons have unit distance. In Pudding SOM the neuron distances are not constrained to integers which enables better space mapping. Therefore the resulting SOM is invariant to the translation and rotation of its structure. Let $\Delta t > 0$ be learning period and the diffusion in continuum will be studied only in discrete-time $t_k = k \cdot \Delta t$, where $k \in \mathbb{N}_0$. The result of SOM learning [30] is the system of weights $\mathbf{w}_i \in \mathbb{R}^n$, where , $i = 1, \ldots, H$ of course. We begin with random weights setting $\mathbf{w}_i(0)$. The weights evolve during learning process and their values in time t_q are denoted as $\mathbf{w}_i(q)$, where $q \in \mathbb{N}_0$. The *Pudding model* is based on the substrate concentrations in neurons and given time. Prepared for SOM learning, we have to study the concentration profile first using single and complete activation procedures.

3.1 Single Activation

The Pudding SOM learning is based on the activation of single neuron. We will study j-th neuron which is supposed to be active in time t_k. Therefore, formally $j = \varphi_k$. The concentration profile in $\mathbb{R}^N$ is depicted on the left part of Fig. 2 for $N = 2$. But it is not necessary to study the substrate concentration in any point. The learning is based only on the concentration (4) in neuron points. The concentration in time t_q is

$$\mathrm{c}(\mathbf{y}, \mathbf{p}_j, t_q - t_k) = \frac{1}{(4\pi D(t_q - t_k))^{N/2}} \cdot \exp\left(-\frac{\|\mathbf{y} - \mathbf{p}_j\|_2^2}{4D(t_q - t_k)}\right) \cdot \exp\left(-\lambda(t_q - t_k)\right) \tag{5}$$

for $q > k$. We can simplify the formula as follows to

$$\mathrm{c}(\mathbf{p}_i, \mathbf{p}_j, t_q - t_k) = \frac{1}{(4\pi D(q-k)\Delta t)^{N/2}} \cdot \exp\left(-\frac{d_{i,j}^2}{4D(q-k)\Delta t}\right) \cdot \exp\left(-\lambda(q-k)\Delta t\right). \tag{6}$$

After the substitution $a = 4D\Delta t > 0$, $b = \lambda \Delta t > 0$ we obtain resulting activation formula

$$\mathrm{c}(\mathbf{p}_i, \mathbf{p}_j, t_q - t_q) = (\pi a(q-k))^{-N/2} \cdot \exp\left(-\frac{d_{i,j}^2}{a(q-k)} - b(q-k)\right). \tag{7}$$

The parameters a, b are dimensionless diffusion and chemical kinetic rates. Both diffusion and chemical reactions occur during nitric oxide activation in the brain. As seen, the diffusion

parameter a is useful for the distance effect. When a is small, the neuron distances $d_{i,j}$ strongly decrease the activation concentration but higher values of a cause only symbolic influence of the distances. The kinetic parameter b is useful for washing out nitric oxide. A small value of b causes the accumulation of nitric oxide after several steps of complete activation will be discussed in the next sections. But higher values of b cause fast washing out with too small activation concentration. Therefore, it is necessary to select the compromise value of both parameters related to neuron distances. When $\min(d_{i,j}) = 1$, then we suggest to use $a = 1, b = 1/10$ for the first experiments as will be demonstrated in next sections.

3.2 Complete Activation

The SOM learning is based on the substrate concentrations in q-th step in time t_q. This concentration is a result of previous activation sequence $\varphi_1, \varphi_2, \ldots \varphi_{q-1}$ using the single activation model (7). Due to the linearity of (1) we can use the additivity principle and directly calculate the cumulative concentration in i-th neuron and step q

$$\begin{aligned} c_{i,q} &= \sum_{k=1}^{q-1} c(\mathbf{p}_i, \mathbf{p}_{\varphi_k}, t_q - t_k) \\ &= \frac{1}{(\pi a)^{N/2}} \cdot \sum_{k=1}^{q-1} \frac{\exp\left(-\frac{d^2_{i,\varphi_k}}{a(q-k)} - b(q-k)\right)}{(q-k)^{N/2}}. \end{aligned} \tag{8}$$

The resulting formula consists of all concentration information that is necessary for SOM learning. Therefore, the concentration $c_{i,q}$ is only a function of activation history, SOM topology and parameters a, b. The impact of parameters a, b is captured in Fig. 1. But history is a result of learning and will be studied in the next section.

The full concentration profile in $2D$ Pudding model after 99 random activation steps ($q = 100$) is depicted on the right part of Fig. 2. Based on previous dimensionless description of diffusion process, we can benefit from the substance concentration profile, and we can formulate the algorithm for diffusive learning in the next section.

From the practical point of view, formula (7) enables pre-calculation for various neuron pairs and time shift $s = q - k$. The second advantage of (7) is in limit behaviour for $s \to +\infty$ when the activation concentration approaches zero value. Let $L \in \mathbb{N}$ be maximum memory size. Practical implementation of complete activation is based on approximation formula

$$c_{i,q} = \sum_{k=\max(1,q-L)}^{q-1} c(\mathbf{p}_i, \mathbf{p}_{\varphi_k}, t_q - t_k). \tag{9}$$

The history length is set to $L \geq 10/b$ to guarantee a low concentration in the omitted steps of the complete activation.

4 Diffusive Learning of SOM

A novel learning algorithm is devoted to Kohonen learning rules [21] as follows. The weight of i-th neuron is changed in q-th step by the following rule

$$\mathbf{w}_i(q) = \mathbf{w}_i(q-1) + \alpha(q) \cdot c_{i,q} \cdot (\mathbf{x}_q - \mathbf{w}_i(q-1)) \tag{10}$$

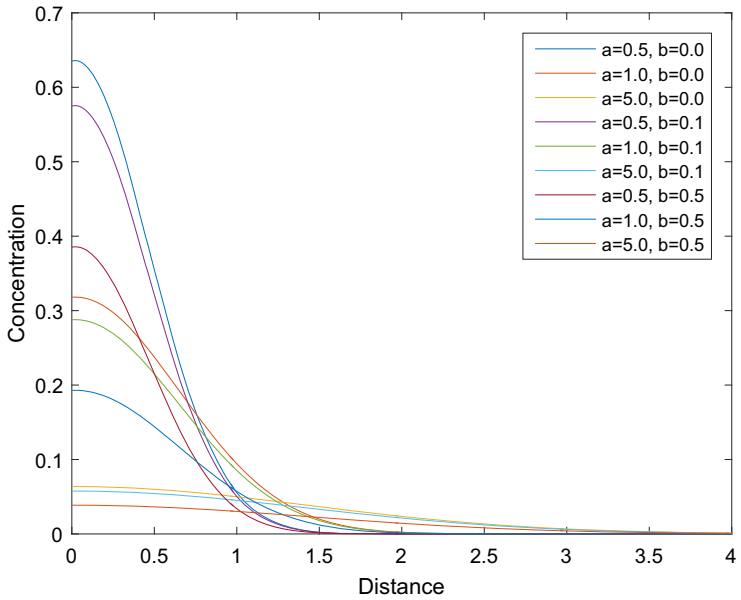


Fig. 1 Learning rate as a function of neuron distance

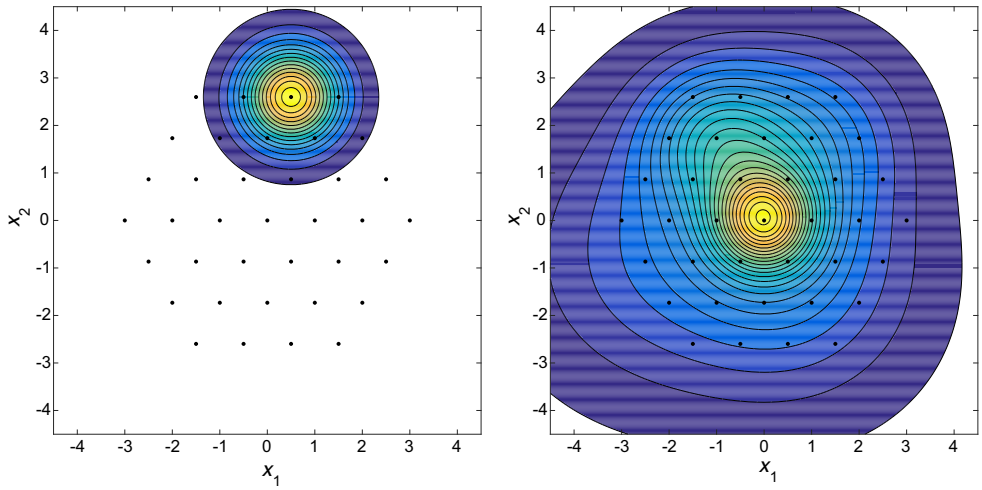


Fig. 2 Concentration profile after single (left) and complete (right) activation ($N = 2$, $a = 1$, $b = 1/10$, $H = 37$, $q = 100$)

for $i = 1, \ldots, H, \mathbf{x}_q \sim \mathrm{U}(\mathcal{S})$ is uniformly selected pattern from $\mathcal{S}$, $c_{i,q}$ is substrate concentration according to (8) and $\alpha(q) > 0$ is ageing function which is supposed to be non-increasing. The winner is also selected according to Kohonen rule [21] as

$$\varphi_q \in \underset{k=1,\ldots,H}{\arg\min} \|\mathbf{x}_q - \mathbf{w}_k\|_2. \tag{11}$$

The main difference between the traditional SOM learning [2] and our approach is in the application of diffusive Eq. (1) which generates the concentration profile (8). The learning feedback is driven by winner index φ_q from (11) which is used in the next step of concentration calculations (8).

As in the traditional SOM learning we have to initialize the weights [2] and use appropriate ageing strategy. We recommend to generate the initial weights from the multivariate Gaussian distribution as

$$\mathbf{w}_i(0) \sim \mathrm{N}(\mathrm{E}\mathbf{X}, \mathrm{var}\mathbf{X}/100) \tag{12}$$

for $i = 1, \ldots, H$. The ageing function $\alpha(q)$ can be constant in the first experiments, but satisfying $\alpha(q) \cdot c_{i,q} \leq 1$ to avoid learning instability.

The time complexity of diffusive SOM initialization is the same as the traditional graph SOM one. But the time complexity of a single learning step is higher. In the case of traditional Kohonen learning the single-step consisting of (11) and (10) uses $2Hn$ multiplications. But the diffusive SOM learning step is extended by cumulative concentrations calculation in H neurons using memory size L. Therefore, the time complexity of the diffusive SOM learning step is $2Hn + HL$.

We will compare the novel method with traditional SOM learning, referred to as Graph SOM, using both artificial and natural pattern sets. Therefore, it is necessary to introduce a set of quality measures for this comparison.

5 Quality Measures

The basic way to design quality measurement is based on measuring distances. The Euclidean distance of points $\mathbf{x}, \mathbf{y}$ in $\mathbb{R}^n$ is denoted $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$.

Using the pattern $\mathbf{x}_j$ we can investigate the distances to weights $\mathbf{w}_k$ and define winner as

$$\mathrm{win}(j) \in \arg\min_{k=1,\ldots,H} \mathrm{d}(\mathbf{x}_j - \mathbf{w}_k) \tag{13}$$

but the function win(j) is of stochastic nature due to possible distance equities. In some cases we found the winner but one i. e. the second winner which is defined as

$$\mathrm{win2}(j) \in \arg\min_{k \in \mathcal{M}_j} \mathrm{d}(\mathbf{x}_j - \mathbf{w}_k), \tag{14}$$

where $\mathcal{M}_j = \{1, \ldots, H\} \backslash \{\mathrm{win}(j)\}$.

By using the system of distances and winners, we can design traditional measures of different nature.

5.1 Distance Penalization

The Quantization Error (QE) is traditionally related to all forms of vector quantization as well as too clustering algorithms [29]. Using linear penalization, we directly penalize the distances between patterns and corresponding winner weights as

$$QE_1 = \sum_{j=1}^{m} \mathrm{d}(\mathbf{x}_j, \mathbf{w}_{\mathrm{win}(j)}). \tag{15}$$

The quadratic penalisation

$$QE_2 = \sum_{j=1}^{m} \mathrm{d}^2(\mathbf{x}_j, \mathbf{w}_{\mathrm{win}(j)}) \tag{16}$$

is also frequently used but has higher sensitivity to outliers.

5.2 Topographic Error

The general topographic rule is: if two objects are close in reality they must be close also in the map. Using this principle the topographic error (TE) [16] is defined as

$$TE = 1 - \frac{1}{m}\sum_{j=1}^{m} g_{\mathrm{win}(j),\mathrm{win2}(j)}, \tag{17}$$

where $\mathbb{G} \in \{0,1\}^{H\times H}$ is SOM topology matrix with $g_{u,v} = \mathrm{I}(\|\mathbf{p}_v - \mathbf{p}_v\|_2 \leq 1)$. The main advantage of TE is in its robustness to outliers. Therefore we use this criterion as main quality measure in this study.

5.3 Correlation Based Measures

The correlations between mutual distances of patterns and mutual distances of winner weights can serve directly as quality measures.

Let i, j be pattern indices. The mutual pattern distances can be defined as $d_{i,j} = \mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)$. The mutual distances of corresponding weights are $\delta_{i,j} = \mathrm{d}(\mathbf{w}_{win(i)}, \mathbf{w}_{win(j)})$.

Finally, we obtain $m(m-1)/2$ pairs of corresponding distances and directly calculate the Pearson correlation coefficient r, Spearmann ρ, or Kendall τ coefficients as a quality measure. The correlation coefficients declared as p-values of independence hypothesis H_0 being comparable with a significant level of 0.05.

5.4 Time Complexity of Measures

The evaluations of QE_1, QE_2 and TE are very fast with the time complexity $\mathrm{O}(mnH)$. The evaluation of correlation measures is a more complex task. The Pearson r has time complexity $\mathrm{O}(mnH + m^2)$ due to simple statistics over $m(m-1)/2$ distance pairs. The Spearmann ρ is complicated with pair sorting and its time complexity is $\mathrm{O}(mnH + m^2\log(m))$. The Kendall τ is not optimal for large pattern sets due to the higher time complexity $\mathrm{O}(mnH + m^4)$. Therefore even the positive correlation between data distances and winner distances corresponds to the general requirement of mapping property between two metric spaces time complexity of its evaluation is the reason to focus on previously mentioned measures.

We will investigate the newly proposed method experimentally in both artificial and nature data case studies. We aim to perform a fair comparison of Pudding SOM and traditional Kohonen learning (Graph SOM) as an efficient tool for data analysis in hexagonal topology. Therefore, we use the hexagonal equidistant topology of Pudding SOM despite the ability of any irregular neuron placing in the case of a novel SOM method. The first case study focuses on the traditional iris flower task.

6 Case Study I: Iris Flower Patterns

We studied different topologies and a various number of neurons for the Pudding SOM and its learning. The results and comparison to traditional SOM were quite comparable for different scenarios. Therefore, as a representative example with high clarity demonstration, we have chosen the following settings. We have selected a hexagonal grid with nineteen neurons placed in 2D space with unit neighborhood distances, i. e. $H = 19$, $N = 2$.

Table 1 Topographical error [%] as function of diffusivity and reaction rate

a	b				
	1/50	1/20	1/10	1/5	1/2
1/3	0.1133	0.0467	0.0933	0.1133	0.1000
1/2	0.0067	0.1133	0.0400	0.0533	0.0267
1	0.0133	0.0400	0.0000	0.0133	0.0267
2	0.0267	0.0267	0.0333	0.0267	0.0400
5	0.0133	0.0133	0.0133	0.0133	0.0133

Table 2 Quality of SOM learning for hexagonal test

Measure	Pudding SOM			Graph SOM		
	Average	Min	Max	Average	Min	Max
QE_1	0.2112	0.2038	0.2224	0.2314	0.2277	0.2519
QE_2	0.2343	0.2051	0.2743	0.2557	0.2335	0.2701
TE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
p-value of r	0.0770	0.0750	0.0841	0.0941	0.0915	0.0998
p-value of ρ	0.0860	0.0779	0.0941	0.1102	0.1021	0.1231
p-value of τ	0.2627	0.2520	0.2701	0.2697	0.2599	0.2902

Artificial two-dimensional data were generated in the first case as follows. The total number of 5000 patterns was generated randomly from seven classes with uniform probability. Origin is a center point of the first class. The centers of the remaining six classes are in the unique distance in the vertices of the hexagon. Individual patterns were generated from this Gaussian mixture with standard deviation $\sigma = 0.2$. The initial weights of Pudding SOM were set according to (12) and ageing factor was set to $\alpha = 0.1$. The role of parameters a, b was studied for memory size $L = 10/b$ to obtain the best setting with a minimum possible topographic error. After 10,000 steps of diffusive learning, we obtained TE included in Table 1. Based on experimental evidence, guaranteed substrate washing out, and therefore short memory effect, we suggest using learning parameters $a = 1, b = 0.1$ adopted in the rest of the study.

Using this setting, we compared Pudding SOM with traditional Kohonen SOM, referred to as Graph SOM. Basic quality measures are included in Table 2 and capture statistics based on 100 launches of the methods. The topographic error is zero in both cases, i.e. both traditional and novel SOM learning maps the data in neurons perfectly. The remaining quality measures differ slightly, and therefore, the Pudding SOM seems to be a good alternative to Kohonen SOM. Resulting weights are depicted in Fig. 3, meanwhile the density map figure (pattern number in given neuron) and traditional U-map [7] are depicted in Fig. 4. Due to the nature of the Pudding model, all the neuron and SOM properties were interpolated on the convex hull of SOM neurons using cubic interpolation. This convention is useful for weight and density interpretation. As seen, the novel algorithm can map the weights proportionally to data coordinates, and corresponding contours are approximately uniformly placed parallel lines in Fig. 3. The density map shows higher central density and six density regions in the network corners. Meanwhile, U-map is approximately constant due to data homogeneity.

The traditional iris flower task [11] originally designed for classifier testing, but we apply it in the case of SOM learning with final class density evaluation. The total number of 150

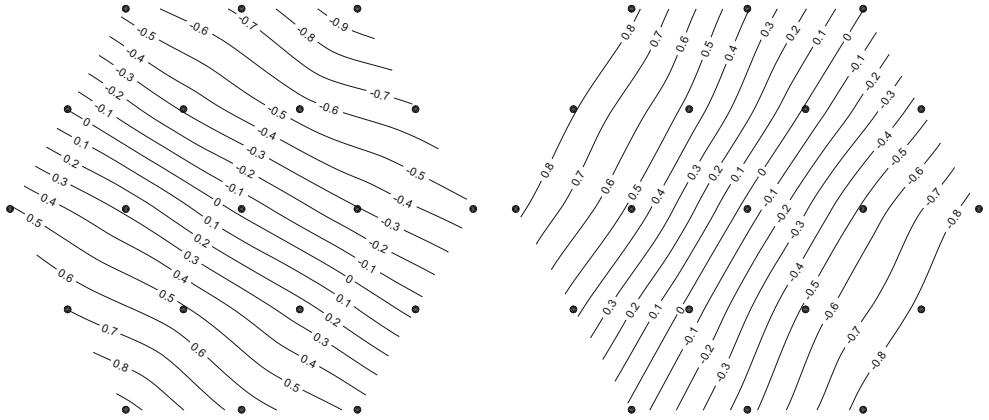


Fig. 3 Resulting weights w_1 (left) and w_2 (right) for hexagonal test

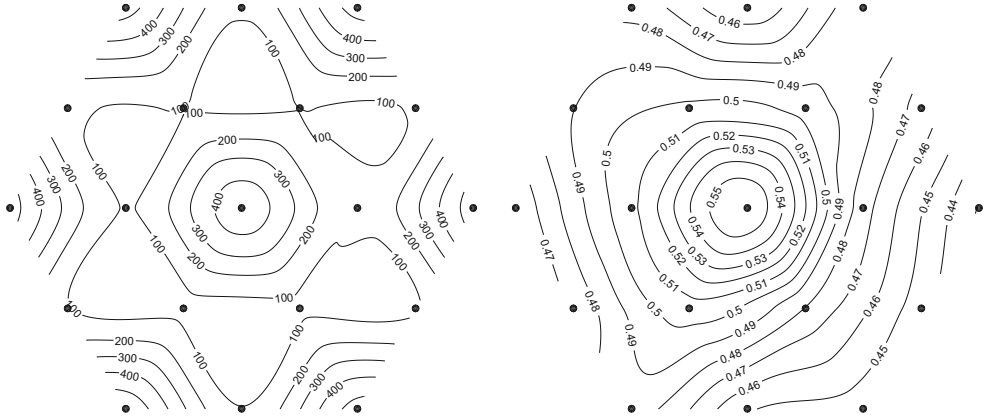


Fig. 4 Density map (left) and U-map (right) for hexagonal test

patterns of three classes (Iris setosa, Iris virginica, Iris versicolor) described by four properties (sepal length, sepal width, petal length, petal width). The initial weights, aging factor, and the number of learning steps were the same as in the previous case. The resulting weight maps are depicted in Fig. 5 as nonlinear mapping of individual iris properties into Pudding SOM. Except for the second property (w_2) containing saddle points, the remaining properties mapped monotonically, which is not the property of Pudding SOM but the property of the given dataset. We can also study class densities as the number of patterns of class insight individual neurons. We can interpolate these properties, and resulting maps are included in Fig. 6 together with the traditional U-map. We can interpolate the Pudding SOM learning results using class membership knowledge. As seen in Fig. 6 the class of Iris setosa is well separated in the left corner, but the remaining two classes are not separable but placed in the opposite part of SOM. We observe iris versicolor species concentrated in the middle and right top, but the iris virginica species placed near the SOM bottom. Patterns do not occupy the remaining part of SOM, maximal values in the U-map confirming that.

The subjective evaluation followed by quality measures evaluation and their comparison with the results of traditional Graph SOM [15] with Kohonen learning, Gaussian characteristic, and following learning strategy. The SOM with $H = 19$ was learned for $E = 9$ with $\alpha = (0.1, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01)$, $R = (5, 3, 3, 1.5, 1, 0.7, 0.5, 0.3, 0.2)$

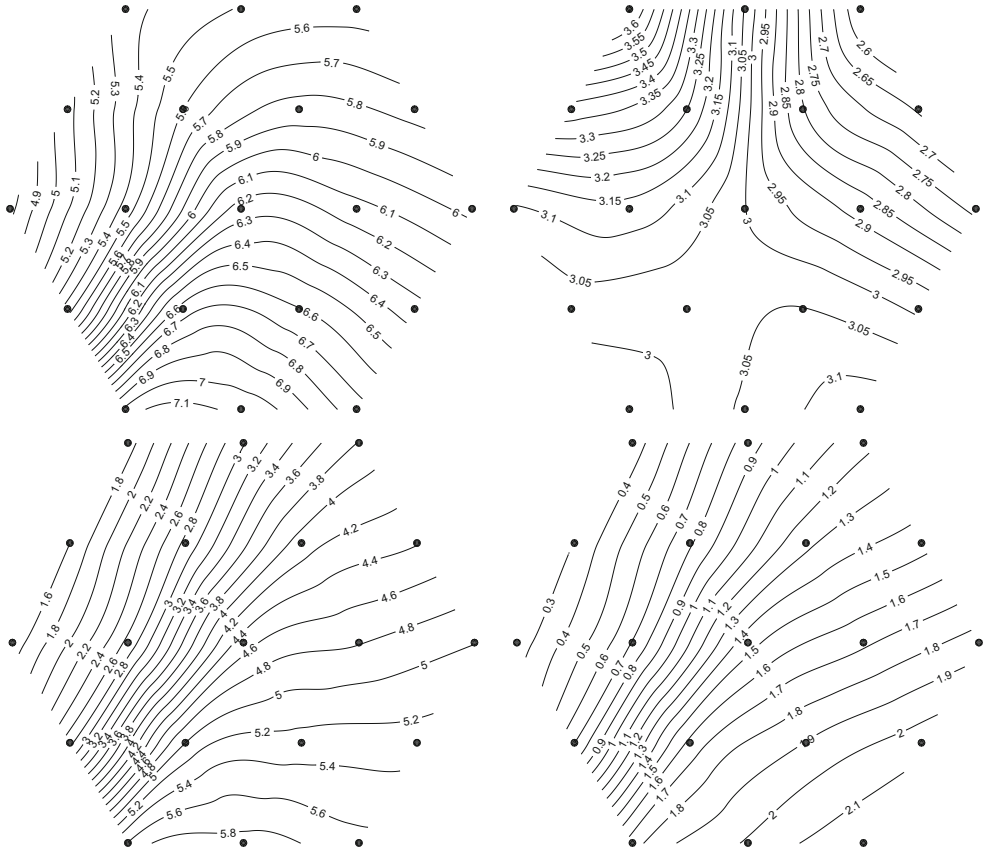


Fig. 5 Resulting weights w_1 (left top), w_2 (right top), w_3 (left bottom), w_4 (right bottom) for iris flowers

and $N_k = 1000$. Table 3 summarize the results. As seen, the topographic error of the novel Pudding SOM is the same as in the traditional case. Therefore, the novel SOM saves the topographic property in a particular case when the neurons are placed equidistantly in hexagonal topology. In other cases, not using the grid equidistant, there would be hardly possible to directly compare Pudding and traditional graph SOM because the Pudding SOM is a more general and complex structure in general. The other quality measures of Pudding SOM are worse but comparable in this case.

This section shows the main properties and advantages of the novel method. The proposed method is fully comparable with the traditional one and enables the straight biological-backed result interpretation. The following section consists of tasks related to macroeconomic indicators and their behavior.

7 Case Study II: Relationships Among Country Economies

After preliminary testing, we applied the Pudding SOM to macroeconomic issues [20] of economic crisis prediction and analysis within thirty-five countries in the world listed in the Table 4. We use the economic indicators as the data input. Data come from Statistical Annex of European Economy presented by the European Commission in autumn 2017 [8]. We study indicators observed in the years 1993–2017. Selected indicators are the total pop-

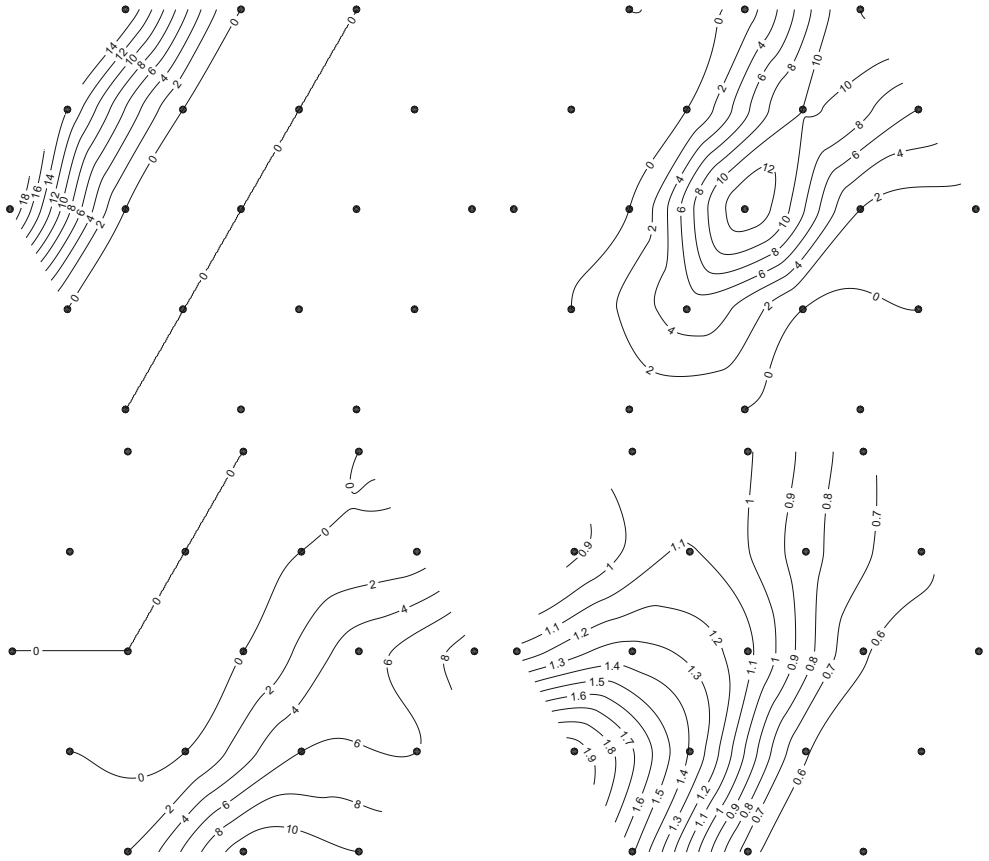


Fig. 6 Resulting class densities—setosa (left top), versicolor (right top), virginica (left bottom) and U-map (right bottom) for iris flowers

Table 3 Quality of SOM learning for iris flower

Measure	Pudding SOM			Graph SOM		
	Average	Min	Max	Average	Min	Max
QE_1	0.4034	0.3839	0.4216	0.3247	0.3119	0.3508
QE_2	0.4546	0.4446	0.4698	0.3596	0.3436	0.3976
TE	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
p-value of r	0.0230	0.0219	0.0256	0.0141	0.0123	0.0201
p-value of ρ	0.0320	0.0299	0.0356	0.0166	0.0158	0.0173
p-value of τ	0.1472	0.1352	0.1559	0.1212	0.1021	0.1397

ulation, unemployment rate, gross domestic product at current market prices, private final consumption expenditure at current prices, gross fixed capital formation at current prices, domestic demand including stocks, exports of goods and services, imports of goods and services and gross national saving. So that nine indicators are monitored in total. We applied logarithmic differences as data preprocessing [19].

Table 4 List of countries

BE	Belgium	MT	Malta	PL	Poland
DE	Germany	NL	Netherlands	RO	Romania
EE	Estonia	AT	Austria	SE	Sweden
IE	Ireland	PT	Portugal	UK	United Kingdom
EL	Greece	SI	Slovenia	MK	Macedonia
ES	Spain	SK	Slovakia	TR	Turkey
FR	France	FI	Finland	ME	Montenegro
IT	Italy	BG	Bulgaria	RS	Serbia
CY	Cyprus	CZ	Czech Republic	AL	Albania
LV	Latvia	DK	Denmark	US	United States
LT	Lithuania	HR	Croatia	JP	Japan
LU	Luxembourg	HU	Hungary		

7.1 Crisis Analysis using Macroeconomic Indicators

In crisis prediction, the individual pattern characterized an object with a unique country, year pair. Therefore, we obtain $35 \times 24 = 840$ patterns from ninth dimensional space of indicators. Performing principal component analysis as necessary preprocessing, we use only two major components saving 67.4% of data variance. The PCA_1 is driven by increasing gross national savings (0.181), increasing gross fixed capital formation (0.052), and decreasing the unemployment rate (−0.046). This PCA_1 saves only 39.3% of variance. Meanwhile, the PCA_2 is driven by increasing the unemployment rate (0.186), decreasing imports of goods and services (−0.087), and increasing gross national savings (0.070). We applied both Pudding SOM and Graph SOM on the data after dimensionality reduction. Being focused on topographic error in this study we obtained $TE = 0.0050$ for Pudding SOM but only $TE = 0.0634$ for Graph SOM. The remaining quality measures are worse but comparable. We present the obtained result in Fig. 7. As seen, the first component has a maximum value in the right bottom corner of SOM and the minimum value at the opposite left top corner. The second component has a maximum value in the left middle corner, but the minimum value is in the right top. The Pudding SOM mapping is monotonic in both cases but nonlinear due to pattern set character. The density map has three significant areas of higher pattern concentration. They are localized in the SOM center, the right top and left middle corners. The U-map has higher values near the left top and left middle corners. It corresponds with a higher gradient of PCA_2 mapping.

The previous PCA and Pudding SOM analysis was based only on macroeconomic indicator changes without prior information about the crisis occur in a given country. We demonstrate the usefulness of Pudding SOM in Fig. 8, where the complete density map is formally split into two parts using the same patterns and previous SOM learning results. The patterns of individual country states as (country, year) pairs for the years up to 2008 declared as before crisis ones, and the rest patterns as after crisis patterns. The patterns before the crisis are localized near the SOM center and right top corner again, but the density in the left middle corner is small. The rest of the patterns after the crisis have different density maps. These patterns are placed in the SOM center again but spread in the right direction. We observe the second-density maximum localized in the left middle corner. The macroeconomic interpretation is straightforward. The majority of countries do not significantly change their

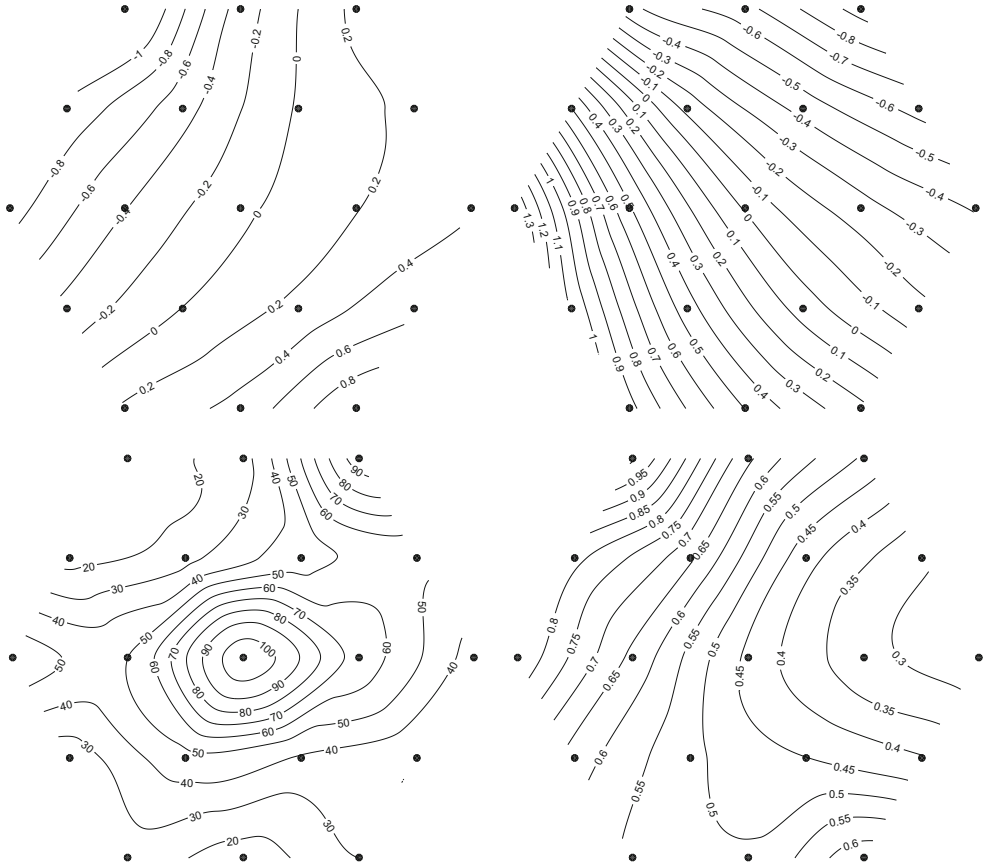


Fig. 7 Resulting weights PCA_1 (left top) and PCA_2 (right top), density (left bottom) and U-map (right bottom) for 9th dimensional macroeconomic patterns

economic indicators after the crisis. But some countries move from the right top corner into the left middle ones, i.e. they change the PCA_2 from minimal to the maximal value. Therefore, some countries had increased the unemployment rate, decreased import rate, and increased gross national savings as a crisis consequence.

7.2 SOM of Country Economies

Using the same data set, we can characterize the country's economy as an object which consists of nine indicators from all years. Therefore we obtained 35 patterns from the 216th-dimensional space of indicator history. After applying the 2D principle component analysis, we get the PCA_1 with 40.8% of the variance, and the first two components save 54.6%. We can interpret the PCA_1 as an adaptivity of the country's economy to novel situations, and the PCA_2 is about the richness of its economy (gross domestic product, gross national savings). The data after dimensionality reduction were analyzed using both Pudding SOM and Graph SOM. Being focused on a topographic error in this study, we obtained $TE = 0.0000$ in both cases. Figure 9 captures the results of Pudding SOM. As seen, the first component has a maximum value at the top of SOM and the minimum value at the bottom. The second component has a maximum value in the right middle corner of SOM and a minimum in

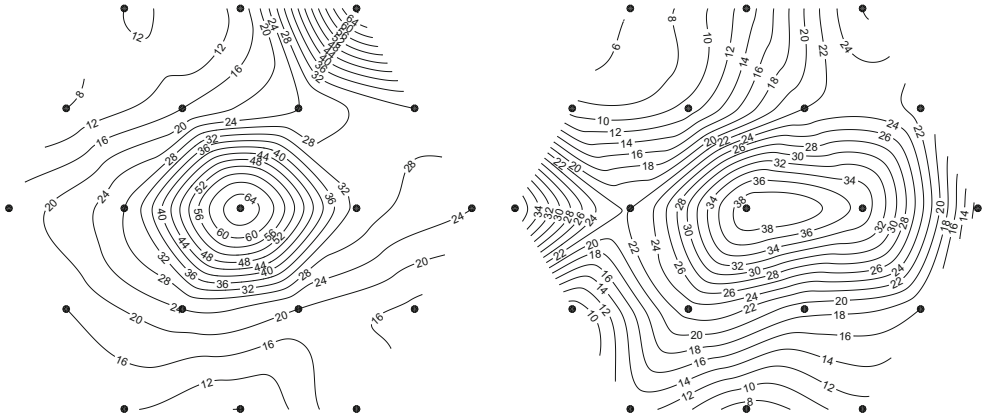


Fig. 8 Resulting class densities of 9th dimensional macroeconomic patterns: before crisis (left) and after crisis (right)

the left middle–bottom part. The density map has a maximum in the right top corner (DE, DK, FR, IT, JP, NL, TR). The lower density is in the bottom part of SOM as a symptom of emptiness. The U-map has a maximum in the SOM bottom, where the country with different behavior (BG) is. We observe the minimum in the right top corner, where the country's economic differences are not significant.

Figure 10 captures the resulting Pudding SOM with country economies labels. The countries are placed not just formally over the SOM due to extreme but not necessarily problematic properties of the Bulgarian economy in a given period. The remaining countries separated by five empty neurons forming a relative compact domain with a spectrum of stable countries (UK, MT, US, etc.) through countries in the middle range (CZ, SK, PL, HU, etc.) to unstable countries (RS, LV, LT, EE, ME). We observe two countries placed in the same neuron or its neighborhood being similar in economic changes, which is a direct macroeconomic interpretation of zero topographical error in this case.

8 Conclusions

We can conclude that the novel structure of the SOM, representing an artificial neuronal network, was successfully developed together with a learning algorithm. It consists of isolated neurons placed in N-dimensional continuum. The neuron connectivity is not explicitly declared, but the neurons communicate via diffusion of nitric oxide in the unconstrained continuum. The resulting algorithm has a physiological, physical, and chemical background, but the processes are simplified to obtain a learning system that is easy to perform. Therefore, the novel learning method has several parameters. The diffusion process is driven by dimensionless diffusion parameter with recommended value $a = 1$ and kinetic parameter with recommended value $b = 0.1$ for equispaced neurons. The learning parameter α has a similar meaning as in the Kohonen approach and can be investigated experimentally as usual. In irregular Pudding SOM topology, the parameters a, b can be set experimentally. The Pudding SOM learning is possible just through a single epoch, i. e. with a constant value of learning parameter α, which is the main advantage comparing multi-epoch Kohonen learning.

After preliminary numerical experiments, we suggest using novel Pudding SOM next to the traditional Kohonen SOM (Graph SOM) not only due to better topographic error

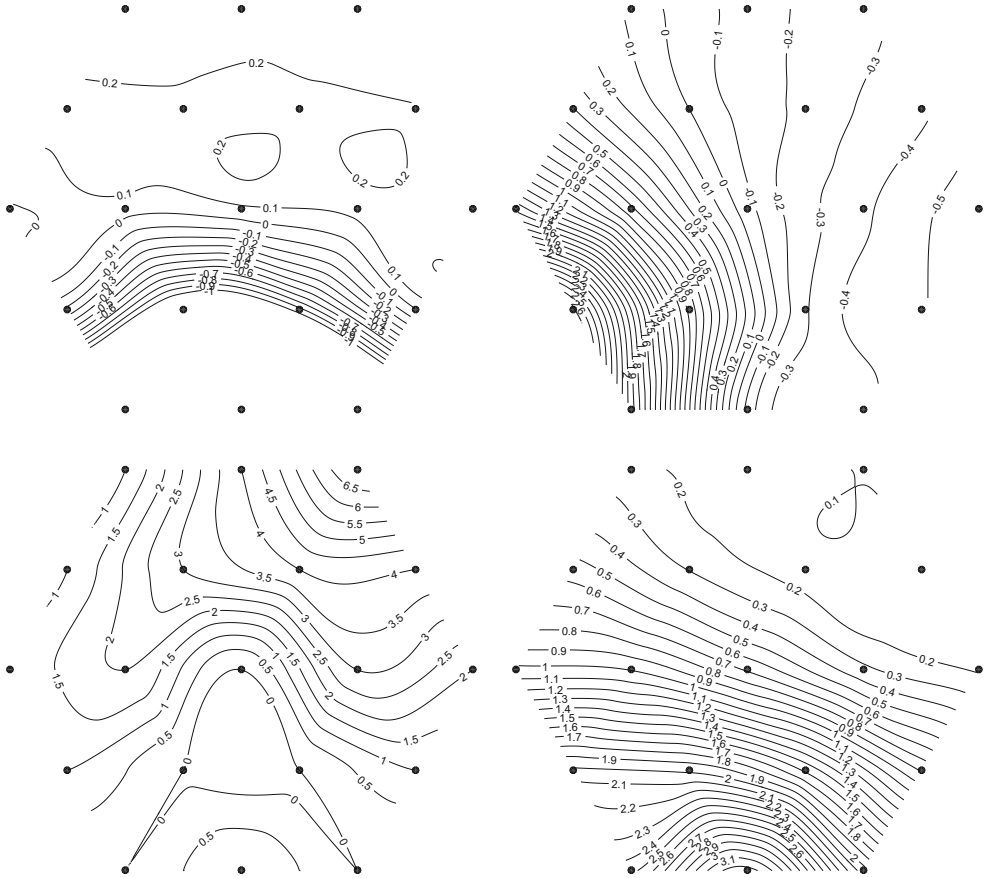


Fig. 9 Resulting weights for map of economies: PCA_1 (left top) and PCA_2 (right top), density (left bottom) and U-map (right bottom) for country classification using 216th dimensional patterns

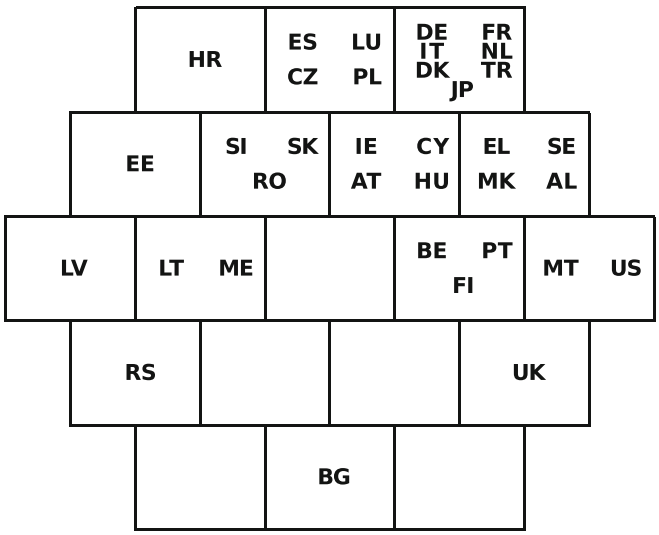


Fig. 10 Resulting map of economies

and comparable quantization errors and correlation coefficients in the case of single epoch learning but mainly due to the biological background of the method. The neurons do not directly interact as in the traditional approach but affect each other via diffusion parameter. Therefore the presented method allows the direct interpretation based on natural processes.

We applied the Pudding SOM to the macroeconomic indicators of several countries. Resulting SOM can indicate both the changes during economic crisis and similarity of country behavior measured by macroeconomic indicator changes.

Acknowledgements The authors would like to acknowledge the support of the research grant SGS20/190/OHK4/3T/14. The second author also acknowledges the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/00-00765 Research Center for Informatics.

Declarations

Conflict of interest All authors declare that they have no conflict of interest in this research.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Abdelsamea MM, Gnecco G, Gaber MM (2017) A SOM-based Chan–Vese model for unsupervised image segmentation. Soft Comput 21(8):2047–2067
2. Alonso E (2010) Computational neuroscience for advancing artificial intelligence: models, methods and applications: models, methods and applications. Premier reference source, Medical Information Science Reference
3. Arik S, Huang T, Lai W, Liu Q (2015) Neural Information processing: 22nd international conference, ICONIP 2015, Istanbul, Turkey, November 9–12, 2015, Proceedings. No. díl 3 in Lecture notes in computer science. Springer, Berlin
4. Brogioli D, Vailati A (2000) Diffusive mass transfer by nonequilibrium fluctuations: Fick's law revisited. Phys Rev E 63:012105
5. Crank J (1975) The mathematics of diffusion/by J. Crank, 2nd edn. Clarendon Press, Oxford
6. Cussler E (2009) Diffusion: mass transfer in fluid systems. Cambridge series in chemical engineering. Cambridge University Press, Cambridge
7. Delgado S, Gonzalo C, Martinez E, Arquero A (2007) Visualizing high-dimensional input data with growing self-organizing maps. Comput Ambient Intell 580–587
8. ECFIN: Statistical annex to european economy. autumn 2017. Tech. rep., European Commission (2017). https://ec.europa.eu/info/files/statistical-annex-european-economy-autumn-2017_en
9. Edelman G, Gally J (1992) Nitric oxide: linking space and time in the brain. Proc Natl Acad Sci 89(24):11651–11652
10. Espenson J (1995) Chemical kinetics and reaction mechanisms. Advanced chemistry series. McGraw-Hill, New York
11. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7(2):179–188
12. Gally JA, Montague PR, Reeke GN, Edelman GM (1990) The no hypothesis: possible effects of a short-lived, rapidly diffusible signal in the development and function of the nervous system. Proc Natl Acad Sci 87(9):3547–3551
13. Garthwaite J, Charles SL, Chess-Williams R (1988) Endothelium-derived relaxing factor release on activation of NMDA receptors suggests role as intercellular messenger in the brain. Nature 336(6197):385–388
14. Gelperin A (1994) Nitric oxide mediates network oscillations of olfactory interneurons in a terrestrial mollusc. Nature 369(6475):61–63
15. Graupe D (2016) Deep learning neural networks: design and case studies. World Scientific Publishing, Singapore

16. Hamel L (2016) Som quality measures: an efficient statistical approach. In: Proceedings of the 11th international workshop WSOM 2016. Springer, Houston, pp 49–59
17. Hartell NA (1996) Strong activation of parallel fibers produces localized calcium transients and a form of ltd that spreads to distant synapses. Neuron 16(3):601–610
18. Hölscher C (1997) Nitric oxide, the enigmatic neuronal messenger: its role in synaptic plasticity. Trends Neurosci 20(7):298–303
19. Hrebik R, Kukal J (2015) The economics and data whitening: aata visualisation. In: Federated conference on software development and object technologies. Springer, Berlin, pp 91–101
20. Hrebik R, Kukal J (2015) Multivarietal data whitening of main trends in economic development. Mathematical methods in economics. University of West Bohemia, Plzeň, pp 279–284
21. Kohonen T (2012) Self-organizing maps. Springer series in information sciences. Springer, Berlin
22. Lancaster JR (1994) Simulation of the diffusion and reaction of endogenously produced nitric oxide. Proc Natl Acad Sci 91(17):8137–8141
23. Lopez PF, Araujo CPS, Baez PG, Martin GS (2003) Diffusion associative network: diffusive hybrid neuromodulation and volume learning. In: International work-conference on artificial neural networks. Springer, Berlin, pp 54–61
24. Lopez PF, Baez PG, Araujo CPS (2015) Nitric oxide diffusion and multi-compartmental systems: modeling and implications. In: International conference on neural information processing. Springer, Berlin, pp 523–531
25. Oja E, Kaski S (1999) Kohonen maps. Elsevier Science, Amsterdam
26. OShea M, Colbert R, Williams L, Dunn S (1998) Nitric oxide compartments in the mushroom bodies of the locust brain. NeuroReport 9(2):333–336
27. Park JH, Straub VA, O'Shea M (1998) Anterograde signaling by nitric oxide: characterization and in vitro reconstitution of an identified nitrergic synapse. J Neurosci 18(14):5463–5476
28. Philippides A, Husbands P, O'Shea M (2000) Four-dimensional neuronal signaling by nitric oxide: a computational analysis. J Neurosci 20(3):1199–1207
29. Pölzlbauer G (2004) Survey and comparison of quality measures for self-organizing maps
30. Rettberg A, Zanella M, Amann M, Keckeisen M, Rammig F (2009) Analysis, architectures and modelling of embedded systems: third IFIP TC 10 international embedded systems symposium, IESS 2009, Langenargen, Germany, September 14–16, 2009, Proceedings. IFIP advances in information and communication technology. Springer, Berlin
31. Senapati M (2006) Advanced engineering chemistry. Laxmi Publications
32. Snyder SH, Bredt DS (1991) Nitric oxide as a neuronal messenger. Trends Pharmacol Sci 12:125–128
33. Thomson JJ (1904) On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. Philos Mag Ser 67(39):237–265
34. Turajlić E (2016) Application of neural networks to denoising of CT images of lungs. In: 2016 XI International symposium on telecommunications (BIHTEL). IEEE, pp 1–6
35. Wood J, Garthwaite J (1994) Models of the diffusional spread of nitric oxide: implications for neural nitric oxide signalling and its pharmacological properties. Neuropharmacology 33(11):1235–1244

CEJOR (2019) 27:161–177
https://doi.org/10.1007/s10100-017-0496-5

ORIGINAL PAPER

Optimal unions of hidden classes

Radek Hrebik[1] · **Jaromir Kukal**[1] · **Josef Jablonsky**[2]

Published online: 19 October 2017

Abstract The cluster analysis is a traditional tool for multi-varietal data processing. Using the k-means method, we can split a pattern set into a given number of clusters. These clusters can be used for the final classification of known output classes. This paper focuses on various approaches that can be used for an optimal union of hidden classes. The resulting tasks include binary programming or convex optimization ones. Another possibility of obtaining hidden classes is designing imperfect classifier system. Novel context out learning approach is also discussed as possibility of using simple classifiers as background of the system of hidden classes which are easy to union to output classes using the optimal algorithm. All these approaches are useful in many applications, including econometric research. There are two main methodologies: supervised and unsupervised learning based on given pattern set with known or unknown output classification. Preferring supervised learning, we can combine the context out learning with optimal union of hidden classes to obtain the final classifier. But if we prefer unsupervised learning, we will begin with cluster analysis or another similar approach to also obtain the hidden class system for future optimal unioning. Therefore, the optimal union algorithm is widely applicable for any kind of classification tasks. The presented techniques are demonstrated on an artificial pattern set and on real data related to crisis prediction based on the clustering of macroeconomic indicators.

Keywords Classification · Cluster analysis · Binary programming · Convex programming · Cluster union · Crisis prediction

✉ Radek Hrebik
Radek.Hrebik@seznam.cz

[1] FNSPE, CTU in Prague, Trojanova 13, 120 00 Prague 2, Czech Republic

[2] FIS, University of Economics, W. Churchill Square 4, 130 67 Prague 3, Czech Republic

1 Introduction

Multidimensional statistical methods represent traditional tools for decision support and pattern classification, e.g., multi-step decision making processes and hierarchical systems. The paper deals with alternative approaches to two-step decision processes that are based on the solution of optimization tasks. The first step of a deterministic decision process can be imperfect, but the second step is designed to enhance the perfectness of the whole system.

The novel methodology is based on cluster analysis, which is commonly used in data mining and statistical data analysis (Santi et al. 2016; Shi 2013). The Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) (Konishi and Kitagawa 2008; DiStefano 2015) can be used to select an optimum number of clusters. The interrelation between two systems of classes is represented by a contingency table (O'Brien 1989), which is frequently used in statistics and displays the frequency of events (Kateri 2014; Harshbarger and Reynolds 2015). Biased estimates of adequate probabilities can be improved by using the Bayesian approach (Wang et al. 2016) for bias reduction. Individual tasks can be solved by binary programming (Weber 1978) or convex programming techniques (Hiriart-Urruty and Lemarechal 1996).

The aim of this research is to design a new method for the optimal union of hidden classes and apply the method to real macroeconomic data. The classification task is formulated in the second section using a contingency table as task support. In the third section the various deterministic approaches to achieve the optimal union of hidden classes are discussed. The original methods can be improved using the Bayesian approach as presented in the fourth section. When a randomized decision is made, we can use mixed strategies as demonstrated in the fifth section. The method of class union is applied to macroeconomic data related to the task of predicting the EU countries crisis in the sixth section which is followed by the concluding remarks.

2 Formulation of the problem

In the case of classification we are well motivated to define the final (output) classes because of their close connection to the solution of the problem. The cardinal question is why to define and use another classes called hidden classes. There is a good analogy with ore mining. The formulation of mining problem is clear. It is necessary to separate the material into two classes: the ore and the residual material. But for a large stone the task is too complex. First, it is necessary to break it into small pieces as symbols of hidden classes and then carefully sort them into two output classes: the ore and the rest by using an effective procedure. Although technical aspects of separation are also useful, they are not discussed here. We will focus on the decision whether given piece of stone is or is not ore. Using majority rule, the pieces with more than fifty per cent of ore belong to the first class (ore). It is a traditional approach but with very low efficiency in general. For example the concentration of gold is very low and therefore no piece of stone would be selected for future gold extraction. This paradox can be easily solved by using cluster sensitivity and critical sensitivity which help to construct more sophisticated strategies not only for ore mining but mainly for general classification of patterns.

Table 1 Contingency table as $\mathbb{F}$

	$\mathscr{H}_1$	$\mathscr{H}_2$	$\mathscr{H}_3$	$\mathscr{H}_4$	$\mathscr{H}_5$
$\mathscr{C}_1$	3	98	7	11	0
$\mathscr{C}_2$	7	4	31	10	1
$\mathscr{C}_3$	1	5	27	9	0

A general classification task distributes m patterns into N classes, and our method is based on preprocessing which places them into H hidden classes. Cluster analysis is a good example of formation of hidden classes. There are many other approaches to performing hidden classification using various kinds of local classifiers, and they are generally imperfect.

Our approach is based on basic characteristics of classification quality which are frequently used in many applications: accuracy (Taylor 1997), class sensitivity (Chang and Slikker 1995) and critical sensitivity (Novakova 2008).

Novel formulation of cluster unioning is based on following notation. The pattern set $\mathscr{S} = \{d_1, \ldots, d_m\}$ is partitioned into a disjoint system of non-empty classes $\mathscr{C}_i \subset \mathscr{S}$ for $i = 1, \ldots, N$. The disjoint system of hidden non-empty groups $\mathscr{H}_j \subset \mathscr{S}$ for $j = 1, \ldots, H$ is also known as the result of imperfect classification. The relation between the classes and the hidden groups is presented via the contingency table $\mathbb{F} \in \mathbb{N}_0^{N \times H}$, where $f_{i,j} = \text{card}\{k : d_k \in \mathscr{C}_i \bigcap \mathscr{H}_j\}$ is the result of pattern counting. Here, $f_{i,j}$ is the number of patterns belonging to both classes $\mathscr{C}_i$ and groups $\mathscr{H}_j$ as joint frequency, which can be relativized as

$$q_{i,j} = \frac{f_{i,j}}{\sum_{k=1}^{H} f_{i,k}}, \tag{1}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, H$.

An example of data partition for three classes and five hidden groups is illustrated in Table 1. The rows represent the output classes, the columns represent hidden classes and the values represent number of patterns as usual in contingency tables. This example of a contingency table will help us recognize the differences among the various unioning strategies.

The paper focuses on the optimal union of hidden classes for the best classification performance using various approaches.

2.1 Other clustering techniques

One of the methods for assessing cluster stability is represented by statistical model of cluster stability which is based on combination of clustering algorithm, yields and estimate of the data partition, namely, the number of clusters (Volkovich et al. 2008). Such approach offers the possibility of evaluating the goodness of a cluster by the similarity amongst the entire cluster and its core. The resemblance among appropriate probability distributions is measured by two-sample tests or by probability distances.

The distances are calculated on clustered samples drawn from the source population according to two different distributions.

One of another approaches offers an algorithm based on self-learning K-means clustering (Volkovich et al. 2013 2013). This method offers an algorithm for simultaneous learning the Mahalanobis like distance and K-means clustering aiming to incorporate data rescaling and clustering, so that the data separability grows iteratively in the rescaled space with its sequential clustering. At each step of the algorithm execution, a global optimization problem is resolved in order to minimize the cluster distortions resting upon the current cluster configuration. The other methodology is based on the probabilistic binomial model of the K-nearest neighbours classification which is the approach based on similarities between the observed and the expected number of neighbours (Volkovich et al. 2011).

Another possible way of research is based on fuzzy approach. The comparison of K-means clustering with fuzzy clustering can be found in Bolin et al. (2014). Hybrid system combining fuzzy clustering and MARS was also discussed as suitable approach for the bankruptcy prediction problem (Andrés et al. 2011). The hybrid model outperformed the other systems, both in terms of the percentage of correct classifications and in terms of the profit generated by the lending decisions.

Another practical use of clustering techniques concerns gene–environment networks under ellipsoidal uncertainty where the functionally related groups of genes and environmental factors are identified by clustering techniques and the corresponding uncertain countries are represented in terms of ellipsoids (Kropat et al. 2010). Other research models in the form of time-continuous and time-discrete dynamics, whose unknown parameters are estimated under constraints on complexity and regularization by various kinds of optimization techniques, ranging from linear, mixed-integer, spline, semi-infinite and robust optimization to conic, e.g., semi-definite programming are also presented in Weber et al. (2011).

Clustering can be useful also in searching algorithm for analysis of time series representing the health trajectories of individuals and Markov models (Ghassempour et al. 2014). The problem is challenging because categorical variables make it difficult to define a meaningful distance between trajectories. The approach is based on hidden Markov models and mapping each trajectory into such model, defining a suitable distance between models and clustering these models with a method based on a distance matrix.

In case of high dimension the cluster analysis faces the problem of overfitting and poor generalization performance and the sheer time taken for conventional algorithms to process large amounts of high-dimensional data. In this case a masked EM algorithm was introduced allowing accurate and time-efficient clustering of up to millions of points in thousands of dimensions (Kadir et al. 2014). The proposed model handles simultaneously the heterogeneity across stock markets and over time, i.e., time-constant and time-varying discrete latent variables capture unobserved heterogeneity among and within stock markets, respectively. The results show a clear distinction between two groups of stock markets, each one characterized by different regime switching dynamics that correspond to different expected return-risk patterns, which is consistent with stylized facts in financial econometrics.

A model-based clustering technique was also introduced as a data analytic tool for financial time series analysis taking into account both time-constant unobserved heterogeneity and hidden regimes within time series using maximum likelihood, a generalization of the BaumWelch algorithm for the hidden Markov models (Dias et al. 2015). The results showed that the statistical methodology performed well in capturing the different regime dynamics of stock markets. It clearly distinguished two groups of countries with different observed patterns.

3 Deterministic case

The hidden classes should be useful in final classification as discussed in previous section. But it is necessary to design the algorithms which will transform the hidden classes into the output ones. The deterministic approach is based on assumption that every hidden class belongs to just one output class. This fact is a kind of mapping which is easy to represent by binary matrix where every column consists of just one unit.

The novel deterministic approach is based on the relationship between hidden groups and the output classes. Our aim is to optimize this relationship as the best mapping from hidden to output classes. Here, strict classifier is defined as mapping

$$\mathrm{c} : \mathscr{L}_H \to \mathscr{L}_N$$

from the set $\mathscr{L}_H$ of hidden class indices to the set $\mathscr{L}_N$ of final class indices, where $\mathscr{L}_n = \{1, \ldots, n\}$. This mapping can be expressed via the matrix

$\mathbb{X} \in \{0, 1\}^{N \times H}$, where $x_{i,j} = 1$ iff $d_k \in \mathscr{H}_j \Rightarrow d_k \in \mathscr{C}_i$. Therefore, $x_{i,j} = 1$ just when for any pattern belonging to $\mathscr{H}_j$ it also belongs to $\mathscr{C}_i$. The uniqueness conditions $\sum_{i=1}^{N} x_{i,j} = 1$ have to be satisfied for $j = 1, \ldots, H$.

There are many quantitative measures for classification efficiency.

First, the accuracy of classification can be expressed as

$$acc = \frac{1}{m} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j} \tag{2}$$

and this will be the subject of maximization.

Using the concept of class sensitivity as a relative frequency of true classification, we can calculate it by the equation

$$se_i = \sum_{j=1}^{H} q_{i,j} x_{i,j} \tag{3}$$

for $i = 1, \ldots, N$.

Average sensitivity can be defined as

$$ase = \frac{1}{N} \sum_{i=1}^{N} se_i. \tag{4}$$

The lower estimate of class sensitivity is defined as critical sensitivity

$$se^* = \min\{se_i : i = 1, \ldots, N\}. \tag{5}$$

We can now formulate several linear programming tasks related to optimum classifier design using the planning matrix $\mathbb{X} \in \{0, 1\}^{N \times H}$, where $x_{i,j} = 1$ indicates $\mathscr{H}_j$ as a part of $\mathscr{C}_i$.

3.1 Accuracy maximization

If we decide to maximize the accuracy of the system without any constraints, the critical sensitivity is frequently very low. To avoid this effect we can constrain the critical sensitivity as follows. Let $s^* \in [0, se^*]$ be the minimum acceptable class sensitivity. Maximizing the accuracy (2) with guaranteed class sensitivities $se_i \geq s^*$ for all classes, we obtain optimization task

$$acc = \frac{1}{M} \sum_{i=1}^{N} \sum_{j=1}^{H} f_{i,j} x_{i,j} = \max \tag{6}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \quad \text{for } j = 1, \ldots, H, \tag{7}$$

$$\sum_{j=1}^{H} q_{i,j} x_{i,j} \geq \mathrm{s}^* \quad \text{for } i = 1, \ldots, N, \tag{8}$$

$$x_{i,j} \in \{0, 1\} \quad \text{for } i = 1, \ldots, N,\ j = 1, \ldots, H, \tag{9}$$

which is a binary programming task (Weber 1978). Having no prior knowledge of the value of se^*, we start with $s^* = 0$. Unfortunately, unbalanced classes can be obtained in this case. Therefore, this method can be efficient only when $s^* > 0$.

In the case of the frequency matrix presented in Table 1, $s^* = 0$ and the accuracy maximization, the class $\mathscr{C}_1$ is formed by $\mathscr{H}_2$ and $\mathscr{H}_4$, the class $\mathscr{C}_2$ is formed by $\mathscr{H}_1$, $\mathscr{H}_3$ and $\mathscr{H}_5$, and the class $\mathscr{C}_3$ is empty. The value of acc was determined to be 0.6916 with $ase = 0.5506$ and $se^* = 0$, and this is the main disadvantage of accuracy maximization without enforcing value of s^*.

3.2 Mean sensitivity maximization

The better performance of classification system can be obtained in the case of mean sensitivity maximization with constrained critical sensitivity as in the previous case. Using the minimum acceptable sensitivity s^* again, we can maximize the mean sensitivity (4) with guaranteed class sensitivities $se_i \geq s^*$ for all classes. Resulting binary optimization task is

$$ase = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{H} q_{i,j}x_{i,j} = \max \tag{10}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \quad \text{for } j = 1,\ldots,H, \tag{11}$$

$$\sum_{j=1}^{H} q_{i,j}x_{i,j} \geq s^* \quad \text{for } i = 1,\ldots,N, \tag{12}$$

$$x_{i,j} \in \{0,1\} \quad \text{for } i = 1,\ldots,N,\ j = 1,\ldots,H. \tag{13}$$

In the case of the frequency matrix presented in Table 1, $s^* = 0$ and the mean sensitivity maximization, the class $\mathscr{C}_1$ is formed by $\mathscr{H}_2$, the class $\mathscr{C}_2$ is formed by $\mathscr{H}_1$ and $\mathscr{H}_5$, the class $\mathscr{C}_3$ is formed by $\mathscr{H}_3$ and $\mathscr{H}_4$. The value of ase was determined to be 0.6105 with $acc = 0.6105$ and $se^* = 0.1509$. The previous two approaches offered a relatively low value of se^*, which is their main disadvantage.

3.3 Maximization of *se**

The inefficiency of previous two approaches motivates us to focus on the maximization of critical sensitivity which is the last but efficient possibility of improving the system of class unioning. The authors suggest this approach as a main tool for optimal unioning of hidden classes. In the case of class equity, we can use a minimax approach and maximize critical sensitivity (5). An adequate non-linear optimization task is

$$se^* = \min\{se_i : i = 1,\ldots,N\} = \max \tag{14}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \quad \text{for } j = 1,\ldots,N, \tag{15}$$

$$x_{i,j} \in \{0,1\} \quad \text{for } i = 1,\ldots,N,\ j = 1,\ldots,H. \tag{16}$$

This task can be converted into a linear optimization task

$$se^+ = \max \tag{17}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \quad \text{for } j = 1,\ldots,H, \tag{18}$$

$$\sum_{j=1}^{H} q_{i,j}x_{i,j} - se^+ \geq 0 \quad \text{for } i = 1,\ldots,N, \tag{19}$$

Table 2 Non-Bayesian approach to class union

Maximization	*acc*	*ase*	se^*	$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{C}_3$
acc	0.6916	0.5506	0.0000	$\mathscr{H}_2 \cup \mathscr{H}_4$	$\mathscr{H}_1 \cup \mathscr{H}_3 \cup \mathscr{H}_5$	
ase	0.6105	0.6105	0.1509	$\mathscr{H}_2$	$\mathscr{H}_1 \cup \mathscr{H}_5$	$\mathscr{H}_3 \cup \mathscr{H}_4$
se^*	0.6105	0.6020	0.3396	$\mathscr{H}_2$	$\mathscr{H}_1 \cup \mathscr{H}_3 \cup \mathscr{H}_5$	$\mathscr{H}_3$

$$x_{i,j} \in \{0, 1\} \quad \text{for } i = 1, \ldots, N,\ j = 1, \ldots, H, \tag{20}$$

$$se^+ \in [0, 1]. \tag{21}$$

Here se^+ is not only value of objective function (14) but also artificial variable (21). The inequalities (19) guarantee that se^+ is a lower bound of critical sensitivity se^* during optimization process and $se^+ = se^*$ in the optimum point.

In the case of the frequency matrix presented in Table 1 and the maximization of se^*, the class $\mathscr{C}_1$ is formed by $\mathscr{H}_2$, the class $\mathscr{C}_2$ is formed by $\mathscr{H}_1$, $\mathscr{H}_4$, and $\mathscr{H}_5$, and the class $\mathscr{C}_3$ is formed by $\mathscr{H}_3$. The value of se^* was determined to be 0.3396 with $acc = 0.6105$ and $ase = 0.6020$. This approach is preferred in the experimental part of our study and can be also used for s^* determination in the *acc* and *ase* maximization tasks.

The results of the previous three approaches are presented in Table 2.

4 Bayesian approach

Respecting the basic principles of mathematical statistic, we use the pattern set as statistical sample but we suppose it will be efficient on the whole statistical ensemble. Therefore, we have to correct relative joint frequencies as characteristics of statistical sample to conditional probabilities which describe the statistical ensemble. The main disadvantage of the relative joint frequencies $q_{i,j}$ lies in their poor statistical properties. They can be interpreted as biased estimates of conditional probability of the patterns belonging to $\mathscr{H}_j$ when belonging to $\mathscr{C}_i$ as

$$p_{i,j} = \text{prob}(d \in \mathscr{H}_j | d \in \mathscr{C}_i). \tag{22}$$

Using a natural Bayesian approach (Wang et al. 2016), we assume uniform prior probabilities (Jaynes 1968) and calculate posterior probabilities as

$$q_{i,j}^{\text{BAY}} = \frac{f_{i,j}^{\text{BAY}}}{\sum_{k=1}^{H} f_{i,k}^{\text{BAY}}}, \tag{23}$$

where

$$f_{i,j}^{\text{BAY}} = f_{i,j} + 1. \tag{24}$$

This approach is preferred in the experimental part of our study as it is an improvement of the previous methods, where $q_{i,j}^{\text{BAY}}$ is used instead of $q_{i,j}$. After applying the

Table 3 Bayesian approach to class union

Maximization	*acc*	*ase*	se^*	$\mathscr{C}_1$	$\mathscr{C}_2$	$\mathscr{C}_3$
acc	0.6681	0.5398	0.0000	$\mathscr{H}_2 \cup \mathscr{H}_4$	$\mathscr{H}_1 \cup \mathscr{H}_3 \cup \mathscr{H}_5$	
ase	0.5931	0.5931	0.1724	$\mathscr{H}_2$	$\mathscr{H}_1 \cup \mathscr{H}_5$	$\mathscr{H}_3 \cup \mathscr{H}_4$
se^*	0.5931	0.5854	0.3621	$\mathscr{H}_2$	$\mathscr{H}_1 \cup \mathscr{H}_3 \cup \mathscr{H}_5$	$\mathscr{H}_3$

Bayesian approach to the illustrative example, we obtain similar results presented in Table 3. The cluster union structure is the same as without Bayesian correction in this case, but not generally.

5 Mixed strategy

There are applications where the stochastic decision process is permitted, i.e., repeated classification of given pattern can generate various class numbers. If permitted, this approach can increase the mean sensitivity of classification. This novel approach is not preferred in this paper but it is included for completeness and as inspiration for future applications.

When we randomize the union of hidden groups $\mathscr{H}_j$, the planning matrix $\mathbb{X} \in [0, 1]^{N \times H}$ consists of probabilities

$$x_{i,j} = \text{prob}(d \in \mathscr{C}_i | d \in \mathscr{H}_j) \tag{25}$$

in which $\mathscr{H}_j$ forms $\mathscr{C}_i$. Using an exponent $\alpha \geq 1$, we can design an optimization task

$$Q = \sum_{i=1}^{N} (1 - se_i)^{\alpha} = \min \tag{26}$$

subject to

$$\sum_{i=1}^{N} x_{i,j} = 1 \qquad \text{for } j = 1, \ldots, N, \tag{27}$$

$$se_i \geq s^* \qquad \text{for } i = 1, \ldots, N, \tag{28}$$

$$0 \leq x_{i,j} \leq 1, \tag{29}$$

which is a convex programming task (Hiriart-Urruty and Lemarechal 1996). This approach is included for completeness and it improves ase maximization when $\alpha = 1$. This optimization task can be converted into a linear programming task when $\alpha = 1$ and when $\alpha \to \infty$. It can also be converted into a quadratic programming task when $\alpha = 2$. Using optimal matrix $\mathbb{X}$, we classify unknown patterns as follows: when classified into $\mathscr{H}_j$ we apply roulette wheel to probabilities $x_{i,j}$ for $i = 1, \ldots, N$ and generate random index i according to these probabilities.

6 Context out classifier system

The algorithm of optimal unions of hidden classes as developed in previous sections is rarely directly applicable to real data. It is necessary to have another algorithm which creates the hidden classes automatically from the pattern features. There are many possibilities of designing such preprocessing. We prefer the parallel system of imperfect classifiers. The imperfect classifier is any system which classifies the patterns to given output classes but with low critical sensitivity. Having a lot of such classifiers we are able to use them for alternative pattern description and forming of hidden classes. As in real life we can focus only on several cases and particular features. The imperfect classifier can be learned in any traditional way of perfect classification but only using several patterns and several properties. This approach is called context out learning here.

We would like to demonstrate the applicability of previous theory in the area of imperfect learning. Having m patterns from N classes where each pattern is a vector from $\mathbb{R}^n$, we can define context out classifier as follows: having original pattern set $\mathbf{X} \in \mathbb{R}^{m\times n}$, $\mathbf{y}^* \in \{1,\ldots,N\}^m$, we select K patterns and L properties randomly to obtain context out sample $\mathbf{X}_{\mathrm{CO}} \in \mathbb{R}^{K\times L}$, $\mathbf{y}^*_{\mathrm{CO}} \in \{1,\ldots,N\}^K$. Therefore, we plan to design context out classifier which makes only imperfect response related to original pattern set. Any traditional classification principle can be used to obtain its response $\mathbf{y} \in \{1,\ldots,N\}^m$ for the complete set.

This imperfect classification procedure can be repeated G-times using Monte Carlo approach. Resulting responses $\mathbf{y}_1,\ldots,\mathbf{y}_G$ form matrix $\mathbf{Y}_G \in \{1,\ldots,N\}^{m\times G}$ where i-th row represents i-th pattern as row vector $\mathbf{r}_i = y_{i,1},\ldots,y_{i,G}$ of imperfect memberships. Only unique columns are saved of course. There are N^G possible values of $\mathbf{r}_i$ but we will focus on unique rows. Their number is constrained as $1 \le H \le \min(m, N^G)$. Obviously, the unique rows represent the hidden classes. Previous theory of hidden class union is therefore directly applicable to any kind of context out classifier system.

6.1 Max margin context out classifier

The abilities of context out classification can be demonstrated as direct application of max margin classifier (Boser et al. 1992). Traditionally, this classifier is applied only for $N = 2$ using bipolar notation as follows: $\mathbf{X}_{\mathrm{CO}} \in \mathbb{R}^{K\times L}$, $\mathbf{y}^*_{\mathrm{CO}} \in \{\pm 1\}^K$ where $+1$ represents $\mathscr{C}_1$ and -1 represents $\mathscr{C}_2$. The bipolar response is driven by formula $y = \mathrm{sign}(w_0 + \sum_{j=1}^{L} w_j x_{CO,j})$ where $\mathbf{y} \in \mathbb{R}^{n+1}$ is unknown vector satisfying $(w_0 + \sum_{j=1}^{L} w_j x_{CO,i,j}) y^*_{CO,i} \ge 1$ for all $i = 1,\ldots,K$ as separability conditions. Using maximum margin condition $\sum_{j=1}^{L} w_j^2 = \min$, the optimum weights are the solution of quadratic optimization task.

The max margin classifier is applicable only to linear separable pattern set. When the context out pattern set does not satisfy this condition we select another one randomly. Our testing example is focused on the case of separable dataset but with non-linear decision rule. Therefore, the separation rule exists but it is non-linear. This testing example will demonstrate the efficiency of imperfect linear classifier for the solution of non-linear classification task.

In our testing example the points close to origin form the first class, i.e. $y = +1$ for $\|\mathbf{x}\|_2 \leq \rho$, while the far points form the second class and $y = -1$ for $\|\mathbf{x}\|_2 \geq R$, where $0 < \rho < R$. The patterns with $\|\mathbf{x}\|_2 \in (\rho, R)$ are prohibited in order to avoid class mixing. The coordinates of individual patterns are generated randomly as $\mathbf{x}_i \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ with eliminated prohibited patterns.

6.2 Testing results

The context out classification approach with embedded max margin method and followed by union of hidden classes has been tested on artificial example according to the non-linear classification task mentioned above. We generate 100 two-dimensional patterns using $\rho = 1$ and $R = 1.1$. We design ten random context out classifiers based on five samples and two properties as a model of complete pattern description in the first case. The results of imperfect learning are depicted in Fig. 1a as switching lines of imperfect classifier. Any plane segment consisting of at least one pattern represents unique hidden class. There are 25 hidden classes in Fig. 1a. Using maximization of critical sensitivity with Bayesian correction, we obtain $se_1 = 0.9500$, $se_2 = 0.9000$ and $se^* = 0.9000$.

Using 20 context out classifiers to previous case, we obtain the better classification systems with $H = 39$, $se_1 = 0.9750$, $se_2 = 0.9667$ and $se^* = 0.9667$, which is depicted in Fig. 1b.

The experiments get more interesting in the case of $L = 1$ when the pattern dimensionality is also reduced. When $m = 100$, $K = 5$, $L = 1$, $G = 10$, we obtain $H = 17$, $se_1 = 0.7750$, $se_2 = 0.7667$ and the imperfect classifiers are depicted in Fig. 1c.

When the number of generation is increased to $G = 20$, we obtain $H = 34$, $se_1 = 0.9250$, $se_2 = 0.9167$ and $se^* = 0.9667$, and the imperfect classifiers are depicted in Fig. 1d.

Using higher values of G, we obtain systems with $se^* = 1.0000$ in previous two cases. But the number of classifiers and hidden classes is too large and resulting system has a low practical importance. The combination of context out learning with imperfect classifiers and hidden class unions is also suitable for solving the linear inseparable problems.

7 Case study: crisis prediction

The new method is demonstrated on real data used for predicting the EU crisis based on macroeconomic indicators. The indicators were evaluated using statistical data from the European Commission (2015). The nine main indicators were selected based on our previous research (Hrebik and Kukal 2015), and these are included in Table 4. Annual data on the 28 EU countries from 1993 to 2017 was processed by logarithmic transformation, and the resulting pattern consisted of nine logarithmic differences in the corresponding indicators. Each country was represented by twenty four patterns, and the maximal values of the differences are shown in Table 5. The main idea for crisis prediction was to perform cluster analysis on the hidden classes according to the pattern properties of each country first. We defined two output classes that present the

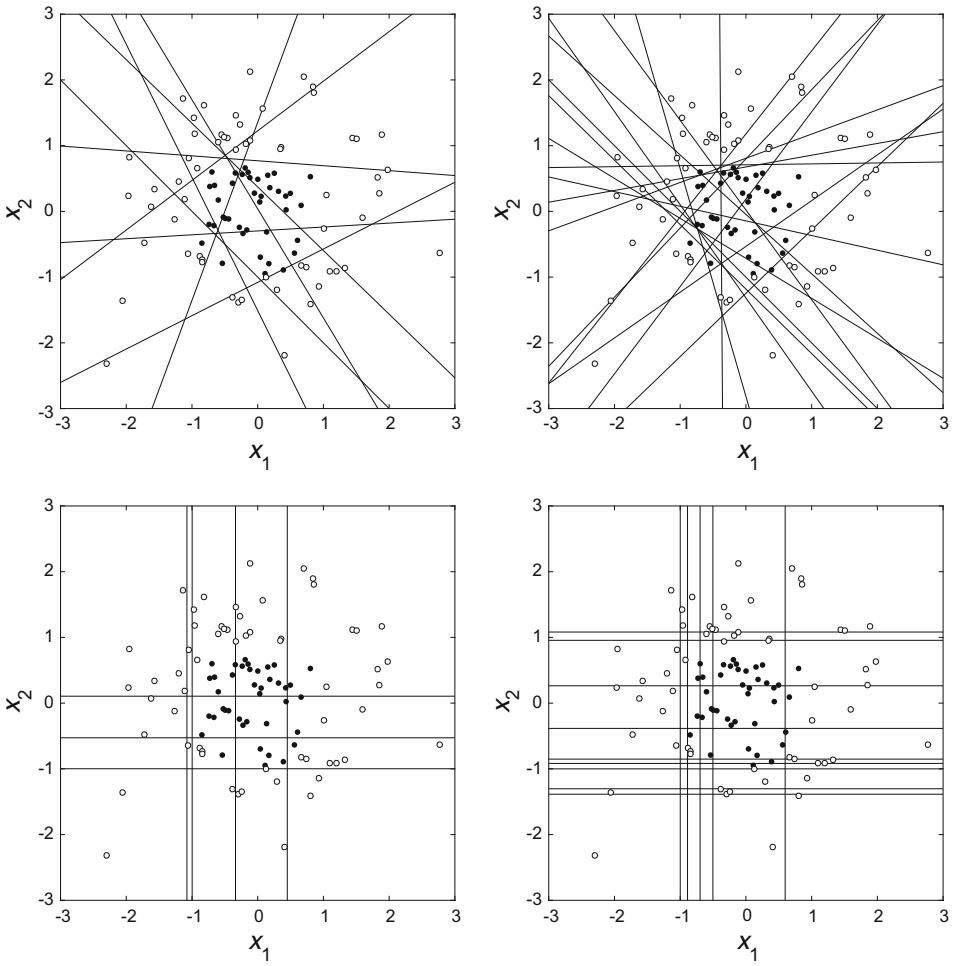


Fig. 1 Context out learning for different parameters. **a** $K = 5$, $L = 2$, $G = 10$, **b** $K = 5$, $L = 2$, $G = 20$, **c** $K = 5$, $L = 1$, $G = 10$, **d** $K = 5$, $L = 1$, $G = 20$

economic indicators before the crisis (1993–2008) and after the crisis (2010–2017). The Optimal union of the hidden classes into the two output classes was the main objective of the following numerical experiments.

We analysed the results of se^* maximization for a number of hidden classes with the value of $H = 2, \ldots, 20$ for each country. This corresponds to 23 patterns per country. The respective number of hidden classes was selected to maximize the critical sensitivity. As an alternative, we used AIC minimization, which also determined the optimal number of hidden classes but without knowledge of the output value. The BIC was not optimal for our purpose because it generated a very low number of classes with small critical sensitivity. The AIC generated similar results as in the case of se^* maximization as demonstrated in Table 6. The pair H_{opt} and se^*_{opt} represents the result of se^* maximization with the knowledge of the output value for each country. By minimizing AIC, we obtained the pair H_{AIC} and se^*_{AIC} without prior knowledge of the crisis status. These two approaches slightly differ in the number of hidden classes but

Table 4 List of descriptors

i	Variable	Explanation
1	TP	Total population
2	UR	Unemployment rate
3	GDP	Gross domestic product at current market prices
4	PFC	Private final consumption expenditure at current prices
5	GFC	Gross fixed capital formation at current prices
6	DD	Domestic demand including stocks at current prices
7	E	Exports of goods and services at current prices
8	I	Imports of goods and services at current prices
9	GNS	Gross national saving

Table 5 Maximal values of absolute logarithmic differences

Country	TP	UR	GDP	PFC	GCF	DD	E	I	GNS
Belgium	0.00869	0.20854	0.07289	0.02172	0.07893	0.03373	0.13982	0.16728	0.18158
Germany	0.00913	0.18805	0.06353	0.03727	0.08158	0.02683	0.14045	0.15143	0.11197
Estonia	0.02148	0.89794	0.34200	0.07250	0.31805	0.08263	0.21123	0.23667	0.23448
Ireland	0.02999	0.62861	0.20250	0.06591	0.17480	0.04896	0.11461	0.11983	0.24273
Greece	0.00828	0.34320	0.09539	0.03934	0.19416	0.02679	0.20830	0.22314	0.59250
Spain	0.01953	0.46000	0.08392	0.01942	0.18369	0.03781	0.13634	0.24476	0.06328
France	0.00750	0.20679	0.05324	0.01815	0.07020	0.01228	0.12833	0.14253	0.14175
Italy	0.00771	0.24201	0.14034	0.01829	0.06863	0.02592	0.18232	0.18160	0.09579
Cyprus	0.02633	0.40968	0.10805	0.09861	0.22839	0.07445	0.09359	0.15101	0.34260
Latvia	0.02123	0.82098	0.50905	0.17530	0.35091	0.14066	0.45519	0.25068	0.87294
Lithuania	0.02253	0.86681	0.45046	0.07411	0.37330	0.09289	0.39897	0.38724	0.29069
Luxembourg	0.02474	0.37949	0.13948	0.07262	0.13778	0.08004	0.12675	0.15509	0.17939
Malta	0.01072	0.14058	0.13630	0.05389	0.23159	0.06185	0.16801	0.17165	0.37205
Netherlands	0.00757	0.26028	0.07916	0.03598	0.07809	0.01561	0.13036	0.13084	0.10970
Austria	0.00967	0.25672	0.07098	0.03019	0.04960	0.02368	0.16962	0.15653	0.14660
Portugal	0.00707	0.20661	0.08123	0.02292	0.15234	0.03711	0.13767	0.18232	0.19307
Slovenia	0.00984	0.29335	0.23349	0.06612	0.19730	0.03993	0.14850	0.20493	0.19259
Slovakia	0.00591	0.26176	0.21145	0.06321	0.22688	0.12197	0.16796	0.18160	0.24914
Finland	0.00488	0.24784	0.16361	0.05873	0.06782	0.02425	0.21706	0.18814	0.18924
Bulgaria	0.02831	0.41522	0.32243	0.19842	0.76461	0.11754	0.32568	0.40767	2.83741
Czech Republic	0.01031	0.42050	0.15415	0.03737	0.13782	0.03815	0.19777	0.15858	0.14781
Denmark	0.00610	0.56798	0.07405	0.03341	0.14477	0.03038	0.14153	0.17680	0.18831
Croatia	0.02888	0.24039	0.34817	0.04841	0.24512	0.05836	0.17959	0.19691	0.34877

Table 5 continued

Country	TP	UR	GDP	PFC	GCF	DD	E	I	GNS
Hungary	0.00523	0.28117	0.17981	0.07504	0.11123	0.06198	0.43393	0.23402	0.28104
Poland	0.00950	0.37013	0.17792	0.03649	0.14505	0.03987	0.14818	0.14680	0.29663
Romania	0.03321	0.44183	0.26711	0.07002	0.38996	0.06283	0.20107	0.21092	0.30956
Sweden	0.02112	0.29171	0.17546	0.05454	0.08589	0.02428	0.11253	0.11692	0.18172
United Kingdom	0.00897	0.30538	0.21514	0.01603	0.11155	0.01298	0.07884	0.08140	0.26905

Table 6 Optimum number of hidden classes for crisis prediction

Country	H_{opt}	se^*_{opt}	H_{AIC}	se^*_{AIC}	se^*_{COL}
Belgium	20	0.8750	20	0.8750	0.8750
Germany	14	0.8750	14	0.8750	0.8750
Estonia	16	0.8750	20	0.8750	0.8750
Ireland	19	0.9333	16	0.8666	0.9333
Greece	15	0.9333	17	0.9333	0.9333
Spain	15	1.0000	20	1.0000	1.0000
France	20	0.8750	18	0.8666	0.8750
Italy	12	0.9333	20	0.9333	0.9333
Cyprus	13	1.0000	20	1.0000	1.0000
Latvia	20	1.0000	20	1.0000	1.0000
Lithuania	19	0.8750	19	0.8750	0.8750
Luxembourg	15	0.9333	19	0.9333	0.8750
Malta	16	0.9333	20	0.9333	0.9333
Netherlands	19	0.9333	20	0.9333	0.9333
Austria	20	0.9333	20	0.9333	0.8750
Portugal	19	1.0000	19	1.0000	1.0000
Slovenia	20	0.9333	19	0.8666	0.9333
Slovakia	20	0.9333	17	0.8750	0.9333
Finland	19	0.8750	19	0.8750	0.8750
Bulgaria	20	1.0000	20	1.0000	1.0000
Czech Republic	20	1.0000	20	1.0000	1.0000
Denmark	9	0.8666	19	0.8666	0.8666
Croatia	17	0.9333	19	0.9333	0.9333
Hungary	15	1.0000	19	1.0000	1.0000
Poland	10	0.9333	19	0.9333	0.9333
Romania	19	1.0000	20	1.0000	1.0000
Sweden	16	0.9333	13	0.8666	0.9333
United Kingdom	19	0.8750	20	0.8666	0.8750

provide very similar values of critical sensitivity. Therefore, crisis prediction can be based on cluster analysis with AIC minimization without loss in the prediction quality.

The same data sets of individual countries were also classified using context out learning as reference method. In the first case we used $G = 20$ random classifiers and the resulting sensitivities se^*_{COL} were very similar to se^*_{OPT} as seen in the Table 6. Therefore, the context out learning also generates efficient prediction system when the number of classifiers is higher.

8 Conclusions

A novel method of optimal cluster union was designed and tested. The main advantage of this approach is the maximization of critical sensitivity or its control at least. The cluster unioning was demonstrated on simple example as efficient but of narrow application range. From the perspective of future applications the method is a bridge between hidden and output classes. There are at least two ways to form the hidden classes: imperfect classification and cluster analysis, which enables to solve more complex task. As in real life the imperfect classification with inefficient decision rule is based on context out approach when only several data patterns are observed and also the set of properties is reduced. Using Monte Carlo approach and the strategy of this context out learning, we obtain alternative pattern description which naturally forms hidden classes. This novel approach was demonstrated on linear inseparable classes with a set of imperfect linear classifiers. When the number of imperfect classifiers is large the sensitivity of cluster unioning is relatively high. The context out learning seems to be inefficient but in the combination with optimal class unions it is a perspective tool for complex decision making.

The second approach combines the cluster unioning with traditional cluster analysis which can be helpful in the case of crisis prediction. An optimal number of clusters was found for each country. We identified three groups of countries as a side effect of our study.

The first group is represented by countries, in which the indicators can easily facilitate crisis prediction. These are the countries with $se^* = 1$, namely Spain, Cyprus, Latvia, Portugal, Bulgaria, the Czech Republic, Hungary, and Romania. The second group is represented by countries, in which it can be more difficult to predict any upcoming crisis. In this case, the value of se^* is lower than 0.875, specifically for Belgium, Germany, Estonia, France, Lithuania, Finland, Denmark, and the United Kingdom. The third group is a compromise between the first and the second group.

The countries in the first group seem to be very sensitive to crisis origins and macroeconomic symptoms, while the sensitivity of the countries forming the second group is significantly lower. Our hypothesis is that they have their own stabilisation mechanism against economic crisis, which can explain the lower predictability of macroeconomic behaviour.

Selection of the hidden class number was primarily based on se^* maximization as left maximum position. A second possible way was to employ the information criteria AIC and BIC. The values of BIC were not recommended because of the small number of clusters with very low values of critical sensitivity. In most cases, AIC suggests

the optimal number of hidden classes with the highest values of critical sensitivity. When AIC was used to select the optimal number of hidden classes, the error of critical sensitivity was up to 0.0667. Using context out learning with twenty imperfect classifiers is efficient alternative to cluster analysis in this case with the error of critical sensitivity up to 0.0583.

Acknowledgements The authors would like to acknowledge the support of the research grants SGS 17/196/OHK4/3T/14 and SGS 17/197/OHK4/3T/14.

References

Andrés JD, Lorca P, de Cos Juez FJ, Sánchez-Lasheras F (2011) Bankruptcy forecasting: a hybrid approach using fuzzy c-means clustering and multivariate adaptive regression splines (mars). Expert Syst Appl 38(3):1866–1875

Bolin JH, Edwards JM, Finch WH, Cassady JC (2014) Applications of cluster analysis to the creation of perfectionism profiles: a comparison of two clustering approaches. Front Psychol 5:343

Boser BE, Guyon IM, Vapnik VN, (1992) A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on computational learning theory, COLT '92, New York, NY, USA, pp 144–152

Chang L, Slikker W (1995) Neurotoxicology: approaches and methods. Elsevier, Amsterdam

Dias JG, Vermunt JK, Ramos S (2015) Clustering financial time series: new insights from an extended hidden Markov model. Eur J Oper Res 243(3):852–864

Directorate General for Economic and Financial Affairs (ECFIN), Statistical annex to European economy. Autumn 2015, Technical report, European Commission (2015). http://ec.europa.eu/economy_finance/publications/eeip/2015-sa-autumn_en.htm

DiStefano J (2015) Dynamic systems biology modeling and simulation. Elsevier, Amsterdam

Ghassempour S, Girosi F, Maeder A (2014) Clustering multivariate time series using hidden Markov models. Int J Environ Res Public Health 11(3):2741–2763

Harshbarger R, Reynolds J (2015) Mathematical applications for the management, life, and social sciences. Cengage Learning, Boston

Hiriart-Urruty J, Lemarechal C (1996) Convex analysis and minimization algorithms I: fundamentals. Springer, Berlin

Hrebik R, Kukal J, (2015) Multivarietal data whitening of main trends in economic development. In: Martincik D, Irgincova J, Janecek P (eds) Mathematical methods in economics, University of West Bohemia, Plzeň, pp 279–284

Jaynes ET (1968) Prior probabilities. IEEE Trans Syst Sci Cybern 4(3):227–241

Kadir SN, Goodman DFM, Harris KD (2014) High-dimensional cluster analysis with the masked em algorithm. Neural Comput 26(11):2379–2394

Kateri M (2014) Contingency table analysis: methods and implementation using R. Statistics for Industry and Technology, Springer, New York

Konishi S, Kitagawa G (2008) Information criteria and statistical modeling. Springer series in statistics. Springer, New York

Kropat E, Weber G-W, Rückmann J-J (2010) Regression analysis for clusters in gene-environment networks based on ellipsoidal calculus and optimization. Dyn Continuous Discrete Impulsive Syst Ser B Appl Algorithms 17(5):639–657

Novakova K (2008) Application of transforms in object recognition (in czech), Ph.D. thesis, FNSPE, CTU in Prague

O'Brien L (1989) The statistical analysis of contingency table designs, concepts and techniques in modern geography, Environmental Publications, University of East Anglia, Norwich

Santi E, Aloise D, Blanchard SJ (2016) A model for clustering data from heterogeneous dissimilarities. Eur J Oper Res 253(3):659–672

Shi G (2013) Data mining and knowledge discovery for geoscientists. Elsevier, Amsterdam

Taylor J (1997) An introduction to error analysis: the study of uncertainties in physical measurements. A series of books in physics. University Science Books, Sausalito

Volkovich Z, Barzily Z, Morozensky L (2008) A statistical model of cluster stability. Pattern Recogn 41(7):2174–2188

Volkovich Z, Barzily Z, Avros R, Toledano-Kitai D (2011) On application of a probabilistic k-nearest neighbors model for cluster validation problem. Commun Stat Theory Methods 40(16):2997–3010

Volkovich Z, Toledano-Kitai D, Weber GW (2013) Self-learning k-means clustering: a global optimization approach (report), J Glob Optim 56 (2):219(14)

Wang J, Ma Y, Ouyang L, Tu Y (2016) A new Bayesian approach to multi-response surface optimization integrating loss function with posterior probability. Eur J Oper Res 249(1):231–237

Weber G (1978) A solution technique for binary integer programming using matchings on graphs. Cornell University, Ithaca

Weber G-W, Defterli O, Gök SZA, Kropat E (2011) Modeling, inference and optimization of regulatory networks based on time series data. Eur J Oper Res 211(1):1–14