POLITECNICO DI TORINO
Repository ISTITUZIONALE

Classification methods based on -logistic models

Original
Classification methods based on -logistic models / Baldi, M.M., Galuzzi, B.G., Messina, E., Kaniadakis, G.. - In: MATHEMATICS AND COMPUTERS IN SIMULATION. - ISSN 0378-4754. - 240:(2026), pp. 347-366. [10.1016/j.matcom.2025.07.001]

Availability:
This version is available at: 11583/3011488 since: 2026-05-27T12:40:08Z

Publisher:
Elsevier

Published
DOI:10.1016/j.matcom.2025.07.001

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 November 2026

Mathematics and Computers in Simulation 240 (2026) 347–366

Contents lists available at ScienceDirect

Mathematics and Computers in Simulation

journal homepage: www.elsevier.com/locate/matcom

Original Articles

Classification methods based on κ-logistic models

Mauro Maria Baldi [a,*], Bruno Giovanni Galuzzi [b], Enza Messina [c], Giorgio Kaniadakis [d,e]

[a] Department of Economics and Law, University of Macerata, Via Crescimbeni 14, Macerata, 62100, Italy
[b] Institute of Bioimaging and Complex Biological Systems, National Research Council (CNR), Via Fratelli Cervi 93, Segrate, 20054, Italy
[c] Department of Informatics, Systems and Communication, University of Milano-Bicocca, Viale Sarca 336, Milano, 20126, Italy
[d] Department of Applied Science and Technology, Politecnico di Torino, Corso Duca degli Abruzzi 24, Torino, 10129, Italy
[e] Institute of Complex Systems, National Research Council (CNR), Corso Duca degli Abruzzi 24, Torino, 10129, Italy

ARTICLE INFO

Keywords:
Logistic regression
κ-exponential function
κ-statistical theory
Classification problems
Machine learning

ABSTRACT

Logistic regression is a simple yet effective technique widely used in machine learning with applications spanning various scientific fields. In this paper, we introduce new logistic regression models based on the κ-exponential function derived from κ-statistical theory, which approaches the standard exponential function as its parameter κ tends to zero. We propose models for both binary and multivariate classification, demonstrating that they extend traditional logistic regression while maintaining the same computational complexity as conventional logistic classifiers. Computational experiments on diverse benchmark data sets show that our κ-logistic classifiers outperform standard logistic regression models in the vast majority of cases.

1. Introduction

Given a matrix $\mathbf{X}$ representing a training set with corresponding target variables $\mathbf{t}$, the aim of a logistic classifier is to predict the class label t of a new input $\mathbf{x}$. The training procedure exploits the sigmoidal function when the number of classes is equal to two (binary case) or the softmax function when the number of classes is greater than two (multiclass case). Both the logistic function and the softmax function make use of the exponential function.

Although simple, this technique is very popular and widespread, as demonstrated by the numerous (though not exhaustive) works cited in the literature review we present in Section 2.

The purpose of this article is to propose new logistic models, named κ-logistic regression, combining logistic regression with the $\exp_\kappa$ function, which is defined as

$$\exp_\kappa(x) = \left(\sqrt{1+\kappa^2x^2}+\kappa x\right)^{\frac{1}{\kappa}}. \tag{1}$$

The motivation for combining logistic classifiers with the $\exp_\kappa$ function is at least twofold. First, the primary reason concerns the popularity of this function. Although it originated in the purely physical context of κ-entropy, based on an approach derived from relativity theory [1,2], its use has had a revolutionary impact in many other fields such as economy [3–6], finance [7–9], computer science [10–12], and epidemiology [13]. Moreover, as shown in [14], the popularity of the $\exp_\kappa$ function is spreading increasingly rapidly. The reason behind the widespread and increasingly growing popularity of the $\exp_\kappa$ function lies in its properties. Among its

* Corresponding author.
E-mail addresses: mauromaria.baldi@unimc.it (M.M. Baldi), brunogiovanni.galuzzi@cnr.it (B.G. Galuzzi), enza.messina@unimib.it (E. Messina), giorgio.kaniadakis@polito.it (G. Kaniadakis).

https://doi.org/10.1016/j.matcom.2025.07.001
Received 6 June 2024; Received in revised form 19 June 2025; Accepted 1 July 2025
Available online 15 July 2025
0378-4754/© 2025 The Authors. Published by Elsevier B.V. on behalf of International Association for Mathematics and Computers in Simulation (IMACS). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

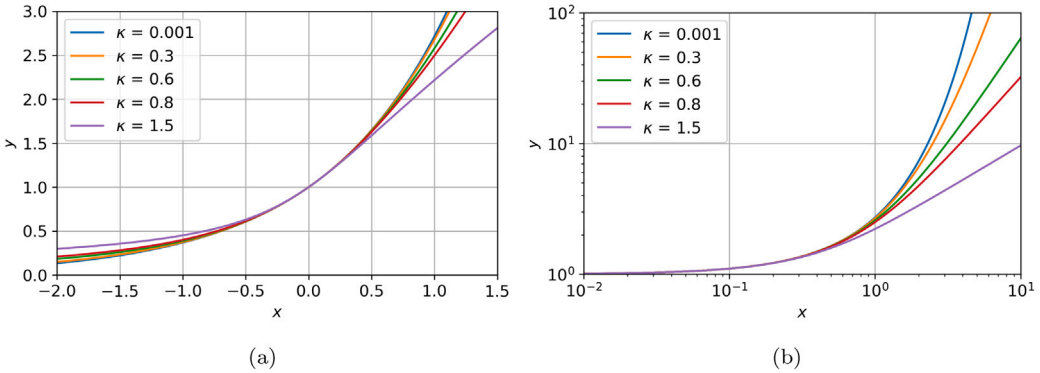

Fig. 1. The $\exp_\kappa$ function on a linear scale (panel a) and a logarithmic scale (panel b), for different values of κ.

most significant properties is the fact that, for all $x \in \mathbb{R}$, $\lim_{\kappa\to 0} \exp_\kappa(x) = \exp(x)$. This property shows us that the $\exp_\kappa$ function is a generalization of the exponential function, and its axiomatic formulation is presented in [15]. But beyond this remarkable property of generalizing the classical exponential function, there is an even more relevant one: its ability to handle rare events. In fact, while the probability distribution of a phenomenon without rare events follows an exponential law, that of a phenomenon with rare events follows a power law.

As shown in [1,2], in addition to the previously mentioned property of the $\exp_\kappa$ function — that in the limit as $\kappa \to 0$ it reduces to the standard exponential function — there is also the following asymptotic behavior:

$$\exp_\kappa(x) \sim |2\kappa x|^{\pm\frac{1}{\kappa}} \quad \text{as } x \to \pm\infty.$$

This double nature of the $\exp_\kappa$ function can be observed in Fig. 1, where we show the $\exp_\kappa$ function for different values of κ.

But it is precisely this dual nature that enables the management of possible rare events through the parameter κ.

Despite the widespread and growing popularity of the $\exp_\kappa$ function, its presence in the field of machine learning is essentially absent. To the best of our knowledge, very little has been done so far in this direction [16]. Therefore, our work represents a pioneering attempt to bring this popular and widely used function into the realm of logistic regression, paving the way for future extensions to other machine learning algorithms.

In this paper, we derive a binary and a multiclass classification model based on $\exp_\kappa$. The validity of our approach is confirmed by computational tests performed on several data sets. We compared our κ-logistic classifiers with their respective pure logistic classifiers and our results were better in the vast majority of cases. Moreover, a comparison showed that our classifiers are competitive with other classifiers in the literature, sometimes even outperforming them.

Furthermore, we prove that the use of the $\exp_\kappa$ function does not alter the complexity of the model. In fact, we show that the computational complexity both for the binary and the multiclass case is the same as for logistic regression. This is not a trivial result, as the introduction of the parameter κ requires considering derivatives with respect to κ during the training procedure.

This paper is organized as follows: in Section 2, we review the state of the art of logistic classifiers with applications and variations proposed in recent years. In Section 3, we introduce the notation for developing the models. In Section 4, we present the model for the binary case, while Section 5 illustrates the model for the multiclass case. In Section 6, we discuss computational complexity of the proposed models. Computational tests are reported in Section 7. We conclude in Section 8.

2. Literature review

Starting from Cox's pioneering work [17], logistic regression has been quickly exploited in many fields including: linguistics [18], medicine [19,20], economy [21,22], business [23,24], natural language processing [25], computer vision [26], bioinformatics [27], neural signal processing [28], and fraud detection [29]. The great diffusion of logistic regression has pushed researchers to refine technicalities and numerical issues in order to provide more efficient variants of the original model. Among these improvements, Owen [30] study the problem of binary logistic regression where one of the two classes is infinitely unbalanced. This problem represents the case where occurrences of one of the two classes are rare events. Jaskie et al. [31] study the problem of logistic regression for positive and unlabeled learning. Shi et al. [32] propose a hybrid algorithm for regularized logistic regression consisting of two phases. The first phase is called iterative shrinkage and it is computationally fast and memory friendly. The second phase is based on a customized interior point method and it is slower but more accurate. Another variant is the logistic stick-breaking problem for non-parametric clustering of general spatially- or temporally-dependent date [33]. Yuan et al. [34] et al. study a relevant problem about logistic regression with regularization. In fact, they show that the coordinate descent method (CDN) [35] for L1-regularized logistic regression becomes inefficient when the loss function is expensive to compute. To overcome this issue, they propose the newGLMNET algorithm, an improvement of the GLMNET Newton-type algorithm by Friedman et al. [36]. Other regularization issues are studied by Wang et al. [37]. Zaidi et al. [38] introduce accelerated logistic regression: a hybrid generative-discriminative

methodology for training logistic regression with high-order features. Foster et al. [39] use improper learning to address online logistic regression. Alternative approaches combine the Choquet integral [40,41] to logistic regression and the Sugeno integral to binary classification [42]. Recently, Kim et al. [43] have studied the relationship between logistic regression and the probabilistic description of fermions in quantum statistical mechanics. More recently, Xavier [44] has proposed hyperbolic regression, among whose objectives is also the improvement of the saturation problem that affects logistic regression problems. Finally, Urbano-Leon et al. [45] have addressed a functional logistic regression model where the observations are not independent.

Compared to the state of the art, our contribution is to introduce two new κ-logistic models (one for the binary and one for the multiclass case) based on the $\exp_\kappa$ function. These models generalize the corresponding logistic models built on the standard exponential function, with the property that as $\kappa \to 0$, our models converge to the ordinary logistic models while maintaining the same computational complexity. Through a series of computational tests, we demonstrate that our models outperform their corresponding logistic models in the vast majority of cases.

3. Preliminary assumptions

Throughout this paper, we are given a training set represented by a matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \end{bmatrix}^T = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,M} \\ x_{2,1} & x_{2,2} & \dots & x_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,M} \end{bmatrix}.$$

consisting of N vectors $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of size M. The training set is also described by a vector $\mathbf{t} = \begin{bmatrix} t_1 & t_2 & \dots & t_N \end{bmatrix}^T$ containing the labels (or target variables) associated to vectors in matrix $\mathbf{X}$. While in regression problems the labels can assume any continuous value, in classification problems the labels are discrete. The main idea is that, given a new input vector $\mathbf{x}$ for which we do not know its target variable t, we can give an estimate $\hat{t}$ of the unknown target variable through a function $y(\mathbf{x}, \mathbf{w})$, where $\mathbf{w} = [w_0, w_1, \dots, w_P]^T$ is a vector of $P+1$ parameters to determine during the training procedure. The prediction for vector $\mathbf{x}$ is given by $\hat{t} = y(\mathbf{x}, \mathbf{w})$. The most general expression for $y(\mathbf{x}, \mathbf{w})$ is given by [26]:

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \tag{2}$$

where $\boldsymbol{\phi}(\mathbf{x}) = \begin{bmatrix} \phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \dots, \phi_P(\mathbf{x}) \end{bmatrix}^T$ is a vector of *basis functions* $\{\phi_j\}_{j=0}^P$ such that $\phi_j : \mathbb{R}^M \to \mathbb{R}$. There are not particular requirements for the basis functions but $\phi_0(\mathbf{x}) = 1 \ \forall \ \mathbf{x} \in \mathbb{R}^M$. The reason for this choice is that we can have an intercept. In fact, if we set $P = M$ and $\phi_j(\mathbf{x}) = x_j \ \forall j \in \{1, \dots, M\}, \forall \mathbf{x} \in \mathbb{R}^M$, then (2) becomes $y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \dots + w_M x_M$ which is one of the most used expressions in logistic regression where w_0 is the intercept. It is also convenient to introduce the following quantities associated to the training set:

$$\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n) = \begin{bmatrix} \phi_0(\mathbf{x}_n) \\ \phi_1(\mathbf{x}_n) \\ \vdots \\ \phi_P(\mathbf{x}_n) \end{bmatrix}, \quad \text{with} \quad n \in \{1, \dots, N\},$$

and

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \boldsymbol{\phi}_2^T \\ \vdots \\ \boldsymbol{\phi}_N^T \end{bmatrix} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \dots & \phi_P(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \dots & \phi_P(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \dots & \phi_P(\mathbf{x}_N) \end{bmatrix},$$

where the matrix $\boldsymbol{\Phi}$ is called the *design matrix*.

At the heart of logistic regression, there is the so-called logistic or sigmoid or sigmoidal function defined as $\sigma(x) = 1/(1+e^{-x})$ that is a differentiable and monotonically increasing function mapping $\mathbb{R}$ onto $(0, 1)$ with $\lim_{x\to-\infty} \sigma(x) = 0$ and $\lim_{x\to+\infty} \sigma(x) = 1$. Therefore, each number can be easily converted into a probability. It is also important to recall an elegant property of the logistic function that is:

$$\frac{d\sigma}{dx}(x) = (1 - \sigma(x))\,\sigma(x) \tag{3}$$

and whose proof can be found in [46]. The idea of the $\exp_\kappa$ function in (1) is to generalize the exponential function e^x such that [1]:

1. $\exp_\kappa(0) = 1$;
2. $\exp_\kappa(x)\,\exp_\kappa(-x) = 1$;
3. $\exp_{-\kappa}(x) = \exp_\kappa(x)$;
4. $\lim_{\kappa\to 0} \exp_\kappa(x) = e^x$;
5. $\exp_\kappa(x) \sim e^x$ for $x \to 0$;
6. $\exp_\kappa(x) \sim |2\kappa x|^{\pm\frac{1}{\kappa}}$ for $x \to \pm\infty$;

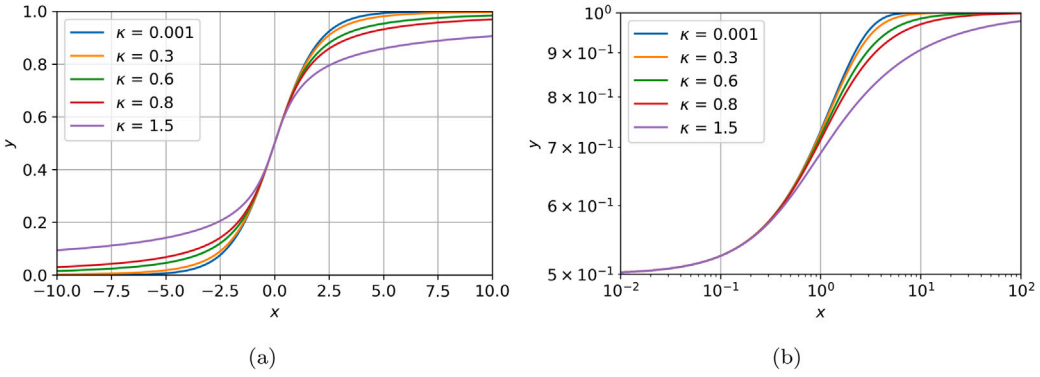

Fig. 2. The σ_κ function on a linear scale (panel a) and a logarithmic scale (panel b), for different values of κ.

7. $\frac{d}{dx}\exp_\kappa(x) = \frac{\exp_\kappa(x)}{\sqrt{1+\kappa^2x^2}}$.

Properties 1. and 2. are in common with the exponential function. Property 3. states that the $\exp_\kappa$ function is even in κ. Therefore, we can just consider non-negative values of parameter κ. Property 4. states that the $\exp_\kappa$ function behaves like the ordinary exponential function when $\kappa \to 0$. Property 5. states that the $\exp_\kappa$ function behaves like the ordinary exponential function when $x \to 0$. Property 6. states that the $\exp_\kappa$ function is asymptotically equivalent to a power function for $x \to \pm\infty$. Property 7. defines the derivative of the $\exp_\kappa$ function. For the purposes of our discussion, it will be necessary to know, among other things, the derivative of the function $\exp_\kappa$ with respect to the parameter κ. Using the technique of logarithmic differentiation, we have:

$$\frac{d}{d\kappa}\exp_k(x) = \frac{\exp_k(x)}{\kappa}\left[\frac{x}{\sqrt{1+\kappa^2x^2}} - \ln\exp_k(x)\right]. \tag{4}$$

We also introduce the inverse function of the $\exp_\kappa$ function, namely the κ-logarithm function defined as follows:

$$\ln_\kappa(x) = \frac{x^\kappa - x^{-\kappa}}{2\kappa}.$$

Like for the $\exp_\kappa$ function, the κ-logarithm function converges to the natural logarithm when κ approaches zero [1]. Moreover, like for the natural logarithm, the κ-logarithm is a strictly monotonically increasing function [2]. Unfortunately, the $\ln_\kappa$ function does not enjoy the property of ordinary logarithms that $\log_a(xy) = \log_a x + \log_a y$ with $a > 0$ and $a \neq 1$.

4. Binary κ-logistic regression

In the binary κ-logistic regression problem, we are given a training set $[\mathbf{X}, \mathbf{t}]$ with $\mathbf{X} \in \mathbb{R}^{N\times M}$ and $\mathbf{t} \in \{0, 1\}^N$. We name C_0 the class corresponding to label 0 and C_1 the class corresponding to label 1. According to Kaniadakis [14], we can define the κ-logistic function mapping $\mathbb{R}$ onto $(0, 1)$ as follows:

$$\sigma_\kappa(x) = \frac{1}{1+\exp_\kappa(-x)}.$$

Fig. 2 shows a graph of the σ_κ function for different values of κ.

Like for the ordinary sigmoidal function, the σ_κ function is suitable to represent a probability because it maps $\mathbb{R}$ onto $(0, 1)$. In fact, we have that $\lim_{x\to-\infty}\sigma_\kappa(x) = 0$ and $\lim_{x\to+\infty}\sigma_\kappa(x) = 1$. Moreover, the function is continuous and its derivative exists anywhere in $\mathbb{R}$. Therefore, given a set of basis functions $\boldsymbol{\phi}$, we can write

$$P(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}\right)$$

and

$$P(C_0|\boldsymbol{\phi}) = 1 - y(\boldsymbol{\phi}) = 1 - \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}\right),$$

where $\mathbf{w}$ represents a vector of parameters whose optimal value can be found through a training procedure. Before showing the procedure to find the optimal values for the parameters $\mathbf{w}$ and κ, we recall a property of the σ_κ function similar to (3): the derivative of the κ-logistic function σ_κ is given by:

$$\frac{d\sigma_\kappa}{dx}(x) = \frac{1}{\sqrt{1+\kappa^2x^2}}\left(1-\sigma_\kappa(x)\right)\sigma_\kappa(x). \tag{5}$$

The proof of (5) can be found in [5].

We wish to point out that, as in [5], we make use of the $\exp_\kappa$ and σ_κ functions. In particular, we exploit property (5). However, our work is significantly different from that of Baldi et al. [5]. In fact, while Baldi et al. [5] study a discrete-time dynamic model of economic growth, we study two machine learning models, respectively for binary and multiclass classification problems.

For the purposes of our future discussion, it is useful to also consider the derivative of the σ_κ function with respect to κ. This derivative can be easily found by using the property stated in (4). Specifically, we obtain:

$$\frac{d}{d\kappa}\sigma_\kappa(x) = \frac{\sigma_\kappa(x)\left(1-\sigma_\kappa(x)\right)}{\kappa}\left[\frac{x}{\sqrt{1+\kappa^2x^2}} + \ln\exp_\kappa(-x)\right]. \tag{6}$$

With these premises, given a set of basis functions $\boldsymbol{\phi}$ and a particular value of the parameter κ, we define y_n as the probability for vector $\mathbf{x}_n$ to belong to class C_1:

$$y_n = P\left(C_1|\mathbf{x}_n, \mathbf{w}, \kappa\right) = \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right).$$

The probability to observe label t_n is given by the following Bernoulli experiment:

$$P(t_n|\mathbf{w}, \kappa) = y_n^{t_n}\left(1-y_n\right)^{1-t_n}.$$

This allows us to introduce the likelihood function defined as follows:

$$P(\mathbf{t}|\mathbf{w}, \kappa) = \prod_{n=1}^{N} y_n^{t_n}\left(1-y_n\right)^{1-t_n}.$$

We can then introduce the following cross-entropy function:

$$E(\mathbf{w}, \kappa) = -\ln P(\mathbf{t}|\mathbf{w}, \kappa) = -\sum_{n=1}^{N}\left(t_n\ln\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right) + (1-t_n)\ln\left(1-\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)\right). \tag{7}$$

The optimal values of the parameters $\mathbf{w}$ and κ can be found minimizing the cross-entropy function. One technique available to minimize the cross entropy $E(\mathbf{w}, \kappa)$ is the gradient method.

We wish to point out that we could have defined the new cross-entropy function as follows:

$$\tilde{E}(\mathbf{w}, \kappa) = -\ln_\kappa P(\mathbf{t}|\mathbf{w}, \kappa).$$

This choice would have been more coherent with the use of the $\exp_\kappa$ function in place of the exponential function. However, the κ-logarithm function can also be written as [1]:

$$\ln_\kappa(x) = \frac{1}{\kappa}\sinh(\kappa\log(x)),$$

which would lead to the following expression for $\tilde{E}(\mathbf{w}, \kappa)$:

$$\tilde{E}(\mathbf{w}, \kappa) = \frac{1}{\kappa}\sinh\left(\kappa E(\mathbf{w}, \kappa)\right)$$

Since $\sinh(x)$ is a monotone function, it follows that:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^P}\tilde{E}(\mathbf{w}, \kappa) = \arg\min_{\mathbf{w}\in\mathbb{R}^P}E(\mathbf{w}, \kappa).$$

Therefore, there is no need to use the κ-logarithm in the new cross-entropy function.

Through (5), we can compute the gradient of the cross-entropy function as shown in Proposition 1.

Proposition 1. *The gradient of the cross-entropy function $E(\mathbf{w}, \kappa)$ w.r.t. the weights $\mathbf{w}$ is:*

$$\nabla_{\mathbf{w}}E(\mathbf{w}, \kappa) = \sum_{n=1}^{N}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}}\left(y_n-t_n\right)\boldsymbol{\phi}_n. \tag{8}$$

Moreover, the partial derivative of the cross-entropy function $E(\mathbf{w}, \kappa)$ w.r.t. the parameter κ is:

$$\frac{\partial}{\partial\kappa}E(\mathbf{w}, \kappa) = \sum_{n=1}^{N}\left[\frac{\mathbf{w}^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} + \ln\exp_\kappa(-\mathbf{w}^T\boldsymbol{\phi}_n)\right]\frac{y_n-t_n}{\kappa}. \tag{9}$$

Proof. We have:

$$\begin{aligned}\nabla_{\mathbf{w}}E(\mathbf{w}, \kappa) &= -\sum_{n=1}^{N}\left(t_n\nabla_{\mathbf{w}}\ln\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right) + (1-t_n)\nabla_{\mathbf{w}}\ln\left(1-\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)\right)\\ &= -\sum_{n=1}^{N}\left(t_n\frac{1}{\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)}\nabla_{\mathbf{w}}\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right) - (1-t_n)\frac{1}{1-\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)}\nabla_{\mathbf{w}}\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)\end{aligned}$$

$$= -\sum_{n=1}^{N} \frac{t_n - \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)}{\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\left(1-\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)} \frac{\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\left(1-\sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} \nabla_{\mathbf{w}}\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)$$

$$= \sum_{n=1}^{N} \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} (y_n - t_n)\boldsymbol{\phi}_n \quad \square$$

The formula for the derivative of cross-entropy with respect to κ is obtained in a completely similar manner, with analogous steps and using (6). We introduce the following matrix $\mathbf{K}$ for expressing (8) in vectorial form:

$$\mathbf{K} = \begin{bmatrix} \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_1\right)^2}} & & & \\ & \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_2\right)^2}} & & \\ & & \ddots & \\ & & & \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_N\right)^2}} \end{bmatrix}.$$

With this new matrix, we can write

$$\nabla_{\mathbf{w}} E(\mathbf{w}, \kappa) = \sum_{n=1}^{N} \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} (y_n - t_n)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^T \mathbf{K}\left(\mathbf{y} - \mathbf{t}\right). \tag{10}$$

Formulae (8) and (10) can be used in the gradient algorithm according to the updating step

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \nabla_{\mathbf{w}} E\left(\mathbf{w}^{(i)}, k^{(i)}\right), \tag{11}$$

$$k^{(i+1)} = k^{(i)} - \eta \frac{\partial}{\partial \kappa} E\left(\mathbf{w}^{(i)}, k^{(i)}\right), \tag{12}$$

where η is a hyperparameter and i denotes the ith epoch of the algorithm. Sometimes, it can be useful not to keep the hyperparameter η constant but to reduce it as the iterations progress. In this case, it is formally more correct to write η_i rather than η, to emphasize the dependency of η on the iteration number. With this premise, (11) and (12) respectively become:

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta_i \nabla_{\mathbf{w}} E\left(\mathbf{w}^{(i)}, k^{(i)}\right), \tag{13}$$

$$k^{(i+1)} = k^{(i)} - \eta_i \frac{\partial}{\partial \kappa} E\left(\mathbf{w}^{(i)}, k^{(i)}\right). \tag{14}$$

However, it is also possible to find a similar expression for the stochastic gradient algorithm. In fact, we can express the cross-entropy function (7) as a sum of partial contributions $E_n(\mathbf{w}, \kappa)$ as follows: $E(\mathbf{w}, \kappa) = \sum_{n=1}^{N} E_n(\mathbf{w}, \kappa)$, where $E_n(\mathbf{w}, \kappa) = -t_n \ln \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right) + (1 - t_n) \ln\left(1 - \sigma_\kappa\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)\right)$. While the gradient algorithm in formulae (8) and (10) evaluates the gradient of the cross-entropy function over the entire training set, the stochastic gradient algorithm computes the partial gradients $\nabla_{\mathbf{w}} E_n(\mathbf{w}, \kappa)$ and $\frac{\partial}{\partial \kappa} E_n(\mathbf{w}, \kappa)$ and then updates the weights and the value of κ for any instance n randomly chosen from the training set at each epoch. Through similar manipulations to those in (8), we can find that

$$\nabla_{\mathbf{w}} E_{\kappa,n}(\mathbf{w}, \kappa) = \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} (y_n - t_n)\boldsymbol{\phi}_n \tag{15}$$

and

$$\frac{\partial}{\partial \kappa} E_{\kappa,n}(\mathbf{w}, \kappa) = \left[\frac{\mathbf{w}^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} + \ln \exp_\kappa(-\mathbf{w}^T\boldsymbol{\phi}_n)\right] \frac{y_n - t_n}{\kappa}. \tag{16}$$

κ-logistic is a generalization of logistic function. Indeeed, in Proposition 2 we show that the formula for the gradient algorithm concerning κ-logistic regression reduces to the corresponding formula for logistic regression when κ approaches 0. However, before proceeding, it is useful to state and prove the following result that will be used in Proposition 2.

Let $\alpha \in \mathbb{R}$. Then,

$$\lim_{\kappa \to 0} \left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} + \frac{1}{\kappa} \ln\left(\sqrt{1+\alpha^2\kappa^2} - \alpha\kappa\right)\right] \frac{1}{\kappa} = 0. \tag{17}$$

This limit can be calculated using the Taylor-Maclaurin series expansions truncated at the third order. Notice that, since α is finite, when κ tends to zero, $\alpha\kappa$ also tends to zero. With this premise, for $\kappa \to 0$, we have:

$$\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} = \alpha - \frac{1}{2}\alpha^3\kappa^2 + o\left(\kappa^3\right) \tag{18}$$

$$\sqrt{1+\alpha^2\kappa^2} - \alpha\kappa = 1 - \alpha\kappa + \frac{1}{2}\alpha^2\kappa^2 + o\left(\kappa^3\right). \tag{19}$$

In light of (19), we have:

$$\ln\left(\sqrt{1+\alpha^2\kappa^2}-\alpha\kappa\right) = -\alpha\kappa + \frac{1}{6}\alpha^3\kappa^3 + o\left(\kappa^3\right). \tag{20}$$

Combining the result of (18) with that of (20), we obtain:

$$\lim_{\kappa\to 0}\left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} + \frac{1}{\kappa}\ln\left(\sqrt{1+\alpha^2\kappa^2}-\alpha\kappa\right)\right]\frac{1}{\kappa} = \lim_{\kappa\to 0}\left(-\frac{1}{3}\alpha^3\kappa + o(\kappa)\right) = 0.$$

Thanks to (17), we have everything we need to prove in Proposition 2 that binary κ-logistic classifiers approach logistic classifiers as κ approaches zero.

Proposition 2. *The gradient vector for binary κ-logistic reduces to the gradient vector of binary logistic regression when $\kappa \to 0$.*

Proof. The gradient for logistic regression is given by the following formula [26]:

$$\nabla E = \boldsymbol{\Phi}^T(\mathbf{y}-\mathbf{t}).$$

First, we compute the following limit: $\lim_{\kappa\to 0}\mathbf{K}$. For those elements which are not on the main diagonal of the matrix, we have $\lim_{\kappa\to 0} 0 = 0$. For the nth element on the main diagonal of the matrix, we have: $\lim_{\kappa\to 0}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} = 1$. Therefore, we have that

$$\lim_{\kappa\to 0}\mathbf{K} = \mathbf{I};$$

where $\mathbf{I}$ stands for the identity matrix of order N. With this premise, we have the following result:

$$\lim_{\kappa\to 0}\nabla_{\mathbf{w}}E(\mathbf{w},\kappa) = \lim_{\kappa\to 0}\boldsymbol{\Phi}^T\mathbf{K}(\mathbf{y}-\mathbf{t}) = \boldsymbol{\Phi}^T\mathbf{I}(\mathbf{y}-\mathbf{t}) = \boldsymbol{\Phi}^T(\mathbf{y}-\mathbf{t}).$$

The proof would not be complete without considering the behavior of the partial derivative of the cross-entropy with respect to κ. This derivative, in fact, is the last component of the gradient of the cross-entropy function. Therefore, it is also necessary to evaluate

$$\lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}E(\mathbf{w},\kappa).$$

We have:

$$\lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}E(\mathbf{w},\kappa) = \lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}\sum_{n=1}^{N}E_n(\mathbf{w},\kappa) = \sum_{n=1}^{N}\lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}E_n(\mathbf{w},\kappa).$$

Therefore, the task leads to calculating

$$\lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}E_n(\mathbf{w},\kappa) = \lim_{\kappa\to 0}\left[\frac{\mathbf{w}^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}^T\boldsymbol{\phi}_n\right)^2}} + \ln\exp_\kappa(-\mathbf{w}^T\boldsymbol{\phi}_n)\right]\frac{y_n-t_n}{\kappa}. \tag{21}$$

We observe first of all that the term $y_n - t_n$ is bounded since $y_n \in (0,1)$ and $t_n \in \{0,1\}$. In particular, $t_n - y_n \in (0,1)$. As previously demonstrated, this limit equals 0. The remaining part of the limit coincides with the limit expressed in (17) in the particular case where $\alpha = \mathbf{w}^T\boldsymbol{\phi}_n$. Therefore, the limit consists of an infinitesimal function multiplied by a bounded function, and hence the result is again zero. Thus,

$$\lim_{\kappa\to 0}\frac{\partial}{\partial\kappa}E(\mathbf{w},\kappa) = 0. \quad \square \tag{22}$$

Apparently, the result of (22) seems out of place compared to the content that we intend to demonstrate in Proposition 2. However, this formula encapsulates a very interesting theoretical meaning. We have indeed shown that the components of the gradient with respect to the parameters $\mathbf{w}$ tend towards the gradient of classical logistic regression, which, however, lacks the parameter κ. On the other hand, if κ tends to zero, it is equivalent to starting the gradient algorithm with $\kappa = 0$. Since the weight update formula includes, among other things, that

$$k^{(i+1)} = k^{(i)} - \eta\frac{\partial}{\partial\kappa}E\left(\mathbf{w}^{(i)}, k^{(i)}\right),$$

then the value of κ is never updated and remains at 0. This, among other things, amounts to having the gradient component with respect to κ of the cross-entropy function always equal to zero, which, therefore, can be omitted. All of this corroborates the fact that a binary κ-logistic classifier tends towards a binary classifier as κ approaches zero. To be precise, there is one last subtlety to point out: we have stated that if κ tends to zero, the gradient descent algorithm should start with the value of κ equal to zero. Technically, however, it is not possible to compute $\exp_\kappa(0)$ due to the exponent being $\frac{1}{\kappa}$. On the other hand, since $\lim_{\kappa\to 0}\exp_\kappa(x) = e^x$, it is possible to continuously extend the function $\exp_\kappa$ in the following way, as done in [5]:

$$\exp_\kappa(x) = \begin{cases}\left(\sqrt{1+\kappa^2x^2}+\kappa x\right)^{\frac{1}{\kappa}} & \text{if } \kappa \neq 0\\ e^x & \text{if } \kappa = 0.\end{cases} \tag{23}$$

Thanks to the generalization proposed in (23), we can confidently start with $\kappa = 0$, in which case a κ-logistic classifier, in light of what has been observed, behaves exactly like a logistic classifier.

To conclude, we provide in Algorithm 1 the pseudocode for training a κ-logistic classifier. For brevity, within the pseudocode, we omit the dependence on certain variables. For example, E_n will stand for $E_n(\mathbf{w}, \kappa)$.

Algorithm 1 Training procedure for binary κ-logistic.

$\mathbf{w} := \mathbf{w}_0$
$\kappa := \kappa_0$
for $it := 1$ to $MAXITER$ **do**
  $E := 0$
  $\nabla_{\mathbf{w}} E := \mathbf{0}$
  $\frac{\partial}{\partial \kappa} E := 0$
  for $n := 1$ to N **do**
    $p_1 := \mathbf{w}^T \boldsymbol{\phi}_n$
    $p_2 := \exp_\kappa(-p_1)$
    $p_3 := \sqrt{1 + \kappa^2 p_1^2}$
    $y_n := \frac{1}{1+p_2}$
    $E_n := -t_n \ln y_n - (1 - t_n) \ln(1 - y_n)$
    $\nabla_{\mathbf{w}} E_n := \frac{1}{p_3}(y_n - t_n)\boldsymbol{\phi}_n$
    $\frac{\partial}{\partial \kappa} E_n := \left[\frac{p_1}{p_3} + \ln p_2\right] \frac{y_n - t_n}{\kappa}$
    $E := E + E_n$
    $\nabla_{\mathbf{w}} E := \nabla_{\mathbf{w}} E + \nabla_{\mathbf{w}} E_n$
    $\frac{\partial}{\partial \kappa} E := \frac{\partial}{\partial \kappa} E + \frac{\partial}{\partial \kappa} E_n$
  end for
  $\mathbf{w} := \mathbf{w} - \eta \nabla_{\mathbf{w}} E$
  $\kappa := \kappa - \eta \frac{\partial}{\partial \kappa} E$
end for

5. Multiclass κ-logistic regression

In this section, we introduce the general case with C classes $C_1, C_2, \ldots, C_C$ and $\mathbf{t} \in \{1, \ldots, C\}^N$. The basic idea is the following: given a new observed point $\mathbf{x}$, we compute $P(C_c|\mathbf{x})$ for each $c \in \{1, \ldots, C\}$ and then assign to vector $\mathbf{x}$ the class with the highest probability. In logistic regression, the so-called *normalized exponential* or *softmax* function $\mathbf{y}$ maps $\mathbb{R}^C$ to $(0, 1)^C$ such that, given vector $\mathbf{a} = [a_1, a_2, \ldots, a_C]^T$, its ith component is defined as

$$y_i = \frac{e^{a_i}}{\sum_{c=1}^{C} e^{a_c}}.$$

An important property of this function is that

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{i,j} - y_j), \tag{24}$$

where $\delta_{i,j}$ is the so-called Kronecker delta function defined as follows:

$$\delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, instead of using label t_n directly, we consider an associated vector $\mathbf{t}_n$ of size $C \times 1$ whose elements are all zero but one equal to 1 in position t_n. It is also useful to define the matrix $\mathbf{T}$ consisting of the collection of vectors $\mathbf{t}_n$ transposed:

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix},$$

which is a matrix of dimensions $N \times C$. We also name $t_{n,c}$ the element of the matrix $\mathbf{T}$ located at the intersection of row n with column c. By construction, we have that

$$\sum_{c=1}^{C} t_{n,c} = 1.$$

In our model, we update the softmax function replacing the exponential function with the $\exp_\kappa$ function and define $\mathbf{z}$ from $\mathbb{R}^C$ to $\mathbb{R}^C$ such that its ith component is:

$$z_i = \frac{\exp_\kappa(a_i)}{\sum_{c=1}^{C} \exp_\kappa(a_c)}.$$

We name this function κ-softmax function. The idea of the κ-softmax function is that if $\mathbf{a} = \left(a_1, \ldots, a_C\right)^T$ is a vector with its ith entry associated to the class C_i, then z_i is the probability of the class C_i having observed the vector $\mathbf{a}$ weighted by the $\exp_\kappa$ function. We are now in a position to prove a property similar to (24), as well as another useful property for handling the training procedure in the case of κ-logistic multiclass classifiers. These properties consist of computing the derivative of z_i with respect to a_j and κ, respectively. Specifically, we have:

$$\begin{aligned}\frac{\partial z_i}{\partial a_j} &= \frac{\frac{1}{\sqrt{1+\kappa^2 a_j^2}}\exp_\kappa(a_j)\,\delta_{i,j}\sum_{c=1}^{C}\exp_\kappa(a_c) - \exp_\kappa(a_i)\frac{1}{\sqrt{1+\kappa^2 a_j^2}}\exp_\kappa(a_j)}{\left(\sum_{c=1}^{C}\exp_\kappa(a_c)\right)^2} \\ &= \frac{1}{\sqrt{1+\kappa^2 a_j^2}}\left(\frac{\exp_\kappa(a_i)}{\sum_{c=1}^{C}\exp_\kappa(a_c)}\delta_{i,j} - \frac{\exp_\kappa(a_i)}{\sum_{c=1}^{C}\exp_\kappa(a_c)}\frac{\exp_\kappa(a_j)}{\sum_{c=1}^{C}\exp_\kappa(a_c)}\right) \\ &= \frac{1}{\sqrt{1+\kappa^2 a_j^2}} z_i(\delta_{i,j} - z_j).\end{aligned} \tag{25}$$

Moreover, to compute the derivative of z_i with respect to κ, we proceed as follows:

$$\frac{\partial}{\partial\kappa} z_i = \frac{\partial}{\partial\kappa}\frac{\exp_\kappa(a_i)}{\sum_{c=1}^{C}\exp_\kappa(a_c)} = \frac{\left(\frac{\partial}{\partial\kappa}\exp_\kappa(a_i)\right)\sum_{c=1}^{C}\exp_\kappa(a_c) - \exp_\kappa(a_i)\sum_{c=1}^{C}\frac{\partial}{\partial\kappa}\exp_\kappa(a_c)}{\left[\sum_{c=1}^{C}\exp_\kappa(a_c)\right]^2}. \tag{26}$$

Applying (4) to (26), after some steps we find

$$\frac{\partial z_i}{\partial\kappa} = \frac{\exp_\kappa(a_i)}{\kappa}\frac{\sum_{c=1}^{C}\left\{\exp_\kappa(a_c)\left[\frac{a_i}{\sqrt{1+\kappa^2 a_i^2}} - \frac{a_c}{\sqrt{1+\kappa^2 a_c^2}} - \ln\frac{\exp_\kappa(a_i)}{\exp_\kappa(a_c)}\right]\right\}}{\left[\sum_{c=1}^{C}\exp_\kappa(a_c)\right]^2}. \tag{27}$$

Then, we replace a_j with $\mathbf{w}_j^T\boldsymbol{\phi}$, where $\mathbf{w}_j$ is a vector of parameters associated to the class $j \in \{1, \ldots, C\}$. Evaluating $\boldsymbol{\phi}$ at $\mathbf{x}_n$ and setting $a_{n,i} = \mathbf{w}_i^T\boldsymbol{\phi}_n$, we can define $z_{n,i,\kappa}$ as the probability of the class C_i having observed the instance $\mathbf{x}_n$ and the parameter κ:

$$z_{n,i,\kappa} = P\left(C_i|\mathbf{x}_n,\ \mathbf{W},\ \kappa\right) = \frac{\exp_\kappa(a_{n,i})}{\sum_{c=1}^{C}\exp_\kappa(a_{n,c})} = \frac{\exp_\kappa\left(\mathbf{w}_i^T\boldsymbol{\phi}_n\right)}{\sum_{c=1}^{C}\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)}, \tag{28}$$

where, for the sake of brevity of notation, $\mathbf{W}$ is a matrix having as rows the vectors $\mathbf{w}_1^T, \ldots, \mathbf{w}_C^T$. Taking into account (25) and using the chain rule for derivatives, we have:

$$\nabla_{\mathbf{w}_j} z_{n,i,\kappa} = \frac{\partial z_{n,i,\kappa}}{\partial\mathbf{w}_j} = \frac{z_{n,i,\kappa}}{\partial a_{n,j}}\frac{\partial a_{n,j}}{\partial\mathbf{w}_j} = \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}} z_{n,i,\kappa}(\delta_{i,j} - z_{n,j})\boldsymbol{\phi}_n. \tag{29}$$

Moreover, taking into account (26), we have:

$$\frac{\partial z_{n,i,\kappa}}{\partial\kappa} = \frac{\exp_\kappa(\mathbf{w}_i^T\boldsymbol{\phi}_n)}{\kappa}\frac{\sum_{c=1}^{C}\left\{\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_i^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_i^T\boldsymbol{\phi}_n\right)^2}} - \frac{\mathbf{w}_c^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)^2}} - \ln\frac{\exp_\kappa\left(\mathbf{w}_i^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)}\right]\right\}}{\left[\sum_{c=1}^{C}\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)\right]^2}. \tag{30}$$

With these premises, we can define the cross-entropy function as follows:

$$E(\mathbf{W},\ \kappa) = -\ln P(\mathbf{T}|\mathbf{w}_1, \ldots, \mathbf{w}_C,\ \kappa) = -\ln\left(\prod_{n=1}^{N}\prod_{c=1}^{C} z_{n,c,\kappa}^{t_{n,c}}\right) = -\sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c}\ln z_{n,c,\kappa}. \tag{31}$$

In Proposition 3, we derive the corresponding formula for computing the gradient of the new cross-entropy function (31).

Proposition 3. *The gradient of (31) with respect to $\mathbf{w}_j$ is:*

$$\nabla_{\mathbf{w}_j} E(\mathbf{W},\ \kappa) = \sum_{n=1}^{N}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}(z_{n,j} - t_{n,j})\boldsymbol{\phi}_n. \tag{32}$$

Moreover,

$$\frac{\partial}{\partial\kappa} E(\mathbf{W},\ \kappa) = -\frac{1}{\kappa}\sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c}\frac{\sum_{h=1}^{C}\left\{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_c^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)^2}} - \frac{\mathbf{w}_h^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)^2}} - \ln\frac{\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}\right]\right\}}{\sum_{h=1}^{C}\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}. \tag{33}$$

Proof. We have that

$$\begin{aligned}\nabla_{\mathbf{w}_j} E(\mathbf{W},\,\kappa) &= -\sum_{n=1}^{N}\sum_{c=1}^{C} t_{n,c}\,\frac{1}{z_{n,c}}\,\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}\,z_{n,c}(\delta_{c,j}-z_{n,j})\boldsymbol{\phi}_n\\ &= \sum_{n=1}^{N}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}\left(z_{n,j}\sum_{c=1}^{C}t_{n,c}-\sum_{c=1}^{C}t_{n,c}\,\delta_{c,j}\right)\boldsymbol{\phi}_n\\ &= \sum_{n=1}^{N}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}(z_{n,j}-t_{n,j})\boldsymbol{\phi}_n.\end{aligned}$$

We also have:

$$\frac{\partial}{\partial\kappa}E(\mathbf{W},\,\kappa) = -\frac{\partial}{\partial\kappa}\sum_{n=1}^{N}\sum_{c=1}^{C}t_{n,c}\ln z_{n,c,\kappa} = -\sum_{n=1}^{N}\sum_{c=1}^{C}t_{n,c}\,\frac{1}{z_{n,c,\kappa}}\,\frac{\partial}{\partial\kappa}z_{n,c,\kappa}. \tag{34}$$

Plugging (30) into (34), we obtain the desired result. □

At this point, we want to make an observation that will prove to be fundamental in demonstrating the computational complexity in Section 6. Indeed, (33) can be significantly simplified by considering that, by construction, only one value of $t_{n,c}$ is equal to 1, while all others are zero. The value of the index c for which $t_{n,c}$ is equal to 1 is given by t_n. With this premise, we can rewrite (33) as follows:

$$\frac{\partial}{\partial\kappa}E(\mathbf{W},\,\kappa) = -\frac{1}{\kappa}\sum_{n=1}^{N}\frac{\sum_{h=1}^{C}\left\{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n\right)^2}}-\frac{\mathbf{w}_h^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)^2}}-\ln\frac{\exp_\kappa\left(\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}\right]\right\}}{\sum_{h=1}^{C}\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}. \tag{35}$$

In a fashion similar to the binary case, in Proposition 4 we prove that the multiclass κ-logistic approaches the multiclass logistic when κ tends to zero.

Proposition 4. *The multiclass κ-logistic approaches the multiclass logistic when κ tends to zero.*

Proof. First, we define $y_{n,i}$ as the ith element of the softmax function evaluated in $\mathbf{w}_i^T\boldsymbol{\phi}_n$:

$$y_{n,i} = \frac{e^{\mathbf{w}_i^T\boldsymbol{\phi}_n}}{\sum_{c=1}^{C}e^{\mathbf{w}_{i,c}^T\boldsymbol{\phi}_n}}.$$

We compute the following limit:

$$\lim_{\kappa\to0}z_{n,j,\kappa} = \frac{\lim_{\kappa\to0}\exp_\kappa\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)}{\sum_{c=1}^{C}\lim_{\kappa\to0}\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)} = \frac{e^{\mathbf{w}_j^T\boldsymbol{\phi}_n}}{\sum_{c=1}^{C}e^{\mathbf{w}_c^T\boldsymbol{\phi}_n}} = y_{n,j}.$$

We have:

$$\lim_{\kappa\to0}\nabla_{\mathbf{w}_j}E(\mathbf{W},\,\kappa) = \lim_{\kappa\to0}\sum_{n=1}^{N}\frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}\left(z_{n,j}-t_{n,j}\right)\boldsymbol{\phi}_n = \sum_{n=1}^{N}\left(z_{n,j}-t_{n,j}\right)\boldsymbol{\phi}_n. \tag{36}$$

As in the binary case, we also need to consider the limit as κ approaches zero of $\frac{\partial}{\partial\kappa}E(\mathbf{W},\,\kappa)$. Consider (33), if we can demonstrate that

$$\lim_{\kappa\to0}\frac{1}{\kappa}\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_c^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)^2}}-\frac{\mathbf{w}_h^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)^2}}-\ln\frac{\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}\right] = 0, \tag{37}$$

then it implies that

$$\lim_{\kappa\to0}\frac{\partial}{\partial\kappa}E(\mathbf{W},\,\kappa) = 0.$$

Since $\mathbf{w}_h^T\boldsymbol{\phi}_n$ is bounded, if we rename $\mathbf{w}_c^T\boldsymbol{\phi}_n$ as α and $\mathbf{w}_h^T\boldsymbol{\phi}_n$ as β, then the limit in (37) is equivalent to:

$$\lim_{\kappa\to0}\left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}}-\frac{\beta}{\sqrt{1+\beta^2\kappa^2}}-\ln\frac{\exp_\kappa(\alpha)}{\exp_\kappa(\beta)}\right]\frac{1}{\kappa}. \tag{38}$$

Since $\exp_\kappa(x) = \frac{1}{\exp_\kappa(-x)}$, we can rearrange (38) as follows:

$$\begin{aligned}
&\lim_{\kappa\to 0}\left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} - \frac{\beta}{\sqrt{1+\beta^2\kappa^2}} - \ln\frac{\exp_\kappa(\alpha)}{\exp_\kappa(\beta)}\right]\frac{1}{\kappa}\\
&= \lim_{\kappa\to 0}\left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} - \frac{\beta}{\sqrt{1+\beta^2\kappa^2}} - \ln\frac{\exp_\kappa(-\beta)}{\exp_\kappa(-\alpha)}\right]\frac{1}{\kappa}\\
&= \lim_{\kappa\to 0}\left\{\left[\frac{\alpha}{\sqrt{1+\alpha^2\kappa^2}} - \frac{1}{\kappa}\ln\left(\sqrt{1+\alpha^2\kappa^2}-\alpha\kappa\right)\right]\frac{1}{\kappa} - \left[\frac{\beta}{\sqrt{1+\beta^2\kappa^2}} - \frac{1}{\kappa}\ln\left(\sqrt{1+\beta^2\kappa^2}-\beta\kappa\right)\right]\frac{1}{\kappa}\right\}. \qquad (39)
\end{aligned}$$

In light of Eq. (17), this limit is zero. □

We want to highlight two important properties revealed by Proposition 4. Firstly, the formula (36) reduces to that of the multiclass logistic regression in [26] when κ tends to zero. Furthermore, even in the multiclass case, we observe the result that the partial derivative of the cross-entropy function tends to zero as κ approaches 0. In this case as well, it appears as if the gradient algorithm starts from zero and is never updated, as the mentioned derivative is always zero. Similar conclusions, therefore, apply as in the binary case.

Finally, we can we rewrite (31) as $E(\mathbf{W},\kappa) = \sum_{n=1}^{N} E_n(\mathbf{W},\kappa)$, where $E_n(\mathbf{W},\kappa) = -\sum_{c=1}^{C} t_{n,c}\ln z_{n,c,\kappa}$ is the partial contribution of instance $\mathbf{x}_n$ to the cross entropy. Like in the binary case, this rewriting can be useful in the stochastic gradient algorithm. With a similar procedure to the one presented in Proposition 3, it is possible to find that the partial gradient $\nabla_{\mathbf{w}_j} E_n(\mathbf{W},\kappa)$ is:

$$\nabla_{\mathbf{w}_j} E_n(\mathbf{W},\kappa) = \frac{1}{\sqrt{1+\kappa^2\left(\mathbf{w}_j^T\boldsymbol{\phi}_n\right)^2}}\left(z_{n,j,\kappa} - t_{n,j}\right)\boldsymbol{\phi}_n,$$

and

$$\frac{\partial}{\partial\kappa}E_n(\mathbf{W},\kappa) = -\frac{1}{\kappa}\sum_{c=1}^{C} t_{n,c}\frac{\sum_{h=1}^{C}\left\{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_c^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)^2}} - \frac{\mathbf{w}_h^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)^2}} - \ln\frac{\exp_\kappa\left(\mathbf{w}_c^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}\right]\right\}}{\sum_{h=1}^{C}\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}. \qquad (40)$$

However, taking into account again that only one value of $t_{n,c}$ is equal to one, and all others are zero (with the value of c such that $t_{n,c}=1$ being precisely t_n), we can reformulate (40) as:

$$\frac{\partial}{\partial\kappa}E_n(\mathbf{W},\kappa) = -\frac{1}{\kappa}\frac{\sum_{h=1}^{C}\left\{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)\left[\frac{\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n\right)^2}} - \frac{\mathbf{w}_h^T\boldsymbol{\phi}_n}{\sqrt{1+\kappa^2\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)^2}} - \ln\frac{\exp_\kappa\left(\mathbf{w}_{t_n}^T\boldsymbol{\phi}_n\right)}{\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}\right]\right\}}{\sum_{h=1}^{C}\exp_\kappa\left(\mathbf{w}_h^T\boldsymbol{\phi}_n\right)}. \qquad (41)$$

Finally, in Algorithm 2, we present the pseudocode for training a multiclass κ-logistic classifier with the gradient algorithm.

6. Computational complexity

In this section, we prove that the computational complexity for training our κ-logistic models is the same as that of the corresponding logistic models, i.e., $O(NM)$ for the binary case and $O(NMC)$ for the multivariate case. We recall that $f(x) = O(g(x))$ for $x\to+\infty$ if there exist $x_0\in\mathbb{R}$ and a positive constant γ such that $|f(x)|\le\gamma|g(x)|$ for all $x > x_0$. Before proceeding, it is important to make the following clarification: we assume that the computational complexity for the calculation of $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ is $O(M)$. Of course, the notation $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ is a generalization that implies $\mathbf{w}^T\boldsymbol{\phi}_n$ in the binary case and $\mathbf{w}_c^T\boldsymbol{\phi}_n$ for all $c\in\{1,\dots,C\}$ in the multiclass case. This assumption stems from the fact that, typically, the number of basis functions P is at most equal to the number of features M, and the evaluation of basis functions is $O(P)$. On the other hand, if $P > M$, such a situation would lead to overfitting issues. Assuming, therefore, $P\le M$, and considering the computational complexity for the calculation of $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})$ as $O(P)$, this, in turn, is $O(M)$. It is also worth emphasizing that if this were not the case, it would not affect the validity of what we will demonstrate, as the complexity of logistic and κ-logistic classifiers would then scale to $O(NP)$ in the binary case and $O(NPC)$ in the multiclass case, but would still remain of the same order.

As further confirmation of what has just been stated, consider the most common case of logistic classification where $\phi_0(\mathbf{x}) = 1$ for all $\mathbf{x}\in\mathbb{R}^M$, $P = M$, and $\phi_j(\mathbf{x}) = x_j$ for all $j\in\{1,\dots,M\}$, for all $\mathbf{x}\in\mathbb{R}^M$. In this case, we have $\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) = w_0 + w_1x_1 + \cdots + w_Mx_M$. In the computation of this expression, we have M products and M additions for a total of $2M$ operations, and therefore the complexity is $O(M)$.

In Proposition 5 we prove that the computational complexity for κ-logistic regression with two classes is $O(NM)$, while in Proposition 6 we provide a proof for the multiclass case. In both cases we assume to employ the gradient algorithm.

Proposition 5. *The computational complexity of the gradient algorithm for κ-logistic regression with two classes is $O(NM)$.*

Algorithm 2 Training procedure for multiclass κ-logistic.

1: **for** c:= 1 to C **do**
2: $\mathbf{w}_c$:= $\mathbf{w}_{c,0}$
3: **end for**
4: κ:= κ_0
5: **for** it:= 1 to $MAXITER$ **do**
6: E:= 0
7: **for** c:= 1 to C **do**
8: $\nabla_{\mathbf{w}_c} E$:= $\mathbf{0}$
9: **end for**
10: $\frac{\partial}{\partial \kappa} E$:= 0
11: **for** n:= 1 to N **do**
12: S:= 0
13: **for** c:= 1 to C **do**
14: p_c:= $\mathbf{w}_c^T \boldsymbol{\phi}_n$
15: q_c:= $\exp_\kappa \left(p_c\right)$
16: r_c:= $\sqrt{1+\kappa^2 p_c^2}$
17: S:= $S + q_c$
18: **end for**
19: **for** c:= 1 to C **do**
20: $z_{n,c,\kappa}$:= $\frac{q_c}{S}$
21: **end for**
22: E_n:= $-\ln z_{n,t_n,\kappa}$
23: **for** c:= 1 to C **do**
24: $\nabla_{\mathbf{w}_c} E_n$:= $\frac{1}{r_c}\left(z_{n,c,\kappa} - t_{n,c}\right)\boldsymbol{\phi}_n$
25: **end for**
26: $\frac{\partial}{\partial \kappa} E_n$:= $-\frac{1}{\kappa}\frac{\sum_{h=1}^{C} q_h\left(\frac{p_{t_n}}{r_{t_n}} - \frac{p_h}{r_h} - \ln\frac{q_{t_n}}{q_h}\right)}{S}$
27: E:= $E + E_n$
28: **for** c:= 1 to C **do**
29: $\nabla_{\mathbf{w}_c} E$:= $\nabla_{\mathbf{w}_c} E + \nabla_{\mathbf{w}_c} E_n$
30: **end for**
31: $\frac{\partial}{\partial \kappa} E$:=$\frac{\partial}{\partial \kappa} E + \frac{\partial}{\partial \kappa} E_n$
32: **end for**
33: **for** c:= 1 to C **do**
34: $\mathbf{w}_c$:= $\mathbf{w}_c - \eta \nabla_{\mathbf{w}_c} E$
35: **end for**
36: κ:=$\kappa - \eta \frac{\partial}{\partial \kappa} E$
37: **end for**

Proof. It is sufficient to sum the computational complexity of each instruction in Algorithm 1. Considering a single epoch of the algorithm, the block from line 7 to line 18 is $O(M)$ and is executed N times as it constitutes the for loop over instances in the training set. Therefore, the overall complexity is $O(NM)$. □

Proposition 6. *The computational complexity of the gradient algorithm for κ-logistic regression with more than two classes is $O(NMC)$.*

Proof. It is sufficient to refer to the pseudocode provided in Algorithm 2. In observing, we note that the involved for loops exhibit the following computational complexities:

- lines 1–3: $O(MC)$
- lines 7–9: $O(MC)$
- lines 13–18: $O(MC)$
- lines 19–21: $O(C)$
- lines 23–25: $O(MC)$
- lines 28–30: $O(MC)$

Since the for loop from lines 11 to 32 consists of N iterations, its computational complexity is $O(NMC)$. Considering a single epoch of the gradient descent algorithm and also taking into account the computational complexity of the remaining instructions, we conclude by stating that the overall computational complexity is $O(NMC)$. □

Table 1
Characteristics of the data sets.

Dataset	Instances	Attributes	Classes	Reference
Heart Failure	299	12	2	mis [48]
Hepatitis	155	19	2	mis [49]
Mushroom	8124	22	2	mis [50]
Skin	245 057	3	2	Bhatt and Dhall [51]
Chirrosis	418	17	3	Dickson et al. [52]
Heart disease	303	13	5	Janosi et al. [53]
Wine	178	13	3	Aeberhard and Forina [54]
Wireless	2000	7	4	Bhatt [55]

Table 2
Number of instances per class for each data set.

Data set	Number of instances per class
Heart Failure	C_0: 203, C_1: 96
Hepatitis	C_0: 123, C_1: 32
Mushroom	C_0: 4208, C_1: 3916
Skin	C_0: 50 859, C_1: 194 198
Chirrosis	C_0: 232, C_1: 25, C_2: 161
Heart disease	C_1: 164, C_2: 55, C_3: 36, C_4: 35, C_4: 13
Wine Quality	C_0: 71, C_1: 59, C_2: 48
Wireless	C_1: 500, C_2: 500, C_3: 500, C_4: 500

7. Computational results

In this section, we conduct extensive computational experiments on the κ-logistic classifiers, thoroughly evaluating their performance and behavior across a variety of settings. All the data sets used in this paper are public available on the UCI Machine Learning Repository [47]. We conducted all our computational tests in Python, utilizing the Numpy package, on a workstation equipped with Intel(R) Xeon(R) CPU E3-1245 v5 3.50 GHz 3.50 GHz with 32 GB DDR4 Memory. The codes and data used in this work are publicly available on the GitHub repository https://github.com/maurobaldi/kLogistics.

7.1. The data sets

We tested our model on several data sets available in the UCI Machine Learning Repository [47]. The main characteristics of the data sets are provided in Tables 1 and 2, respectively.

7.2. Performance metrics

To evaluate the performances of the two classifiers, we used accuracy (ACC), precision ($PREC$), recall (REC), and F1-measure ($F1$). To define these performance metrics, it is first necessary to establish the concepts of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in both the binary and multiclass cases. Let $(\mathbf{X}_{TE}, \mathbf{t}_{TE})$ denote a test set consisting of K instances on which the classifiers are tested. Clearly, the number of features (i.e., columns) of the matrix $\mathbf{X}_{TE}$ will be the same as the matrix $\mathbf{X}$ describing the training set. In other words, $\mathbf{X}_{TE}$ is a $K \times M$ matrix and $\mathbf{t}_{TE}$ is a $K \times 1$ vector. We will denote by $\hat{t}_i$ the class predicted by a classifier (binary or multiclass) for the instance $i \in \{1, \dots, K\}$, while t_i will represent the ith label of the test set. With these premises established, we can define the concepts of TP, TN, FP, and FN for both binary and multiclass cases. Regarding the binary case, we have the following definitions:

- TP is the number of instances that are actual positive (i.e., belonging to class 1) and that were classified as positive. More formally:

$$TP = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = 1 \wedge t_i = 1\}\right|.$$

- TN is the number of instances that are actual negative (i.e., belonging to class 0) and that were classified as negative. More formally:

$$TN = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = 0 \wedge t_i = 0\}\right|.$$

- FP is the number of instances that are negative (i.e., belonging to class 0) and that were classified as positive. More formally:

$$FP = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = 1 \wedge t_i = 0\}\right|.$$

- FN is the number of instances that are positive (i.e., belonging to class 1) and that were classified as negative. More formally:

$$FN = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = 0 \wedge t_i = 1\}\right|.$$

Regarding the multiclass case, the same definitions apply to each class $c \in \{1, \dots, C\}$:

- $TP(c)$: the number of true positives for class c, i.e., the number of instances correctly classified as belonging to class c. More formally:

$$TP(c) = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = c \wedge \hat{t}_i = t_i\}\right|.$$

- $TN(c)$: the number of true negatives for class c, i.e., the number of instances correctly classified as not belonging to class c. More formally:

$$TN(c) = \left|\{i \in \{1, \dots, K\} : \hat{t}_i \neq c \wedge \hat{t}_i = t_i\}\right|.$$

- $FP(c)$: the number of false positives for class c, i.e., the number of instances incorrectly classified as belonging to class c. More formally:

$$FP(c) = \left|\{i \in \{1, \dots, K\} : \hat{t}_i = c \wedge \hat{t}_i \neq t_i\}\right|.$$

- $FN(c)$: the number of false negatives for class c, i.e., the number of instances incorrectly classified as not belonging to class c. More formally:

$$FN(c) = \left|\{i \in \{1, \dots, K\} : \hat{t}_i \neq c \wedge t_i = c\}\right|.$$

We briefly recall the definitions of the performance measures starting with the binary case. The accuracy is the ratio of the instances correctly guessed over the total number of instances, namely

$$ACC = \frac{|i \in \{1, \dots, K\} : \hat{t}_i = t_i|}{K}. \tag{42}$$

Precision is the proportion of the positive identifications that were actually correct, namely

$$PREC = \frac{TP}{TP + FP}. \tag{43}$$

The recall is the proportion of actual positives that were correctly identified, namely

$$REC = \frac{TP}{TP + FN}. \tag{44}$$

The F1-measure consists of the harmonic mean between precision and recall:

$$F1 = 2\frac{PREC \times REC}{PREC + REC}. \tag{45}$$

The highest possible value of $F1$ is 1, indicating perfect precision and recall, and the lowest possible value is 0, if either precision or recall is zero.

In a multiclass data set, (42) remains unchanged. However, (43)–(45) cannot be directly applied because they just work with two classes. Consequently, these definitions must be updated according to three criteria: the macro average, the weighted average, and the micro average. The formal definitions of precision, recall, and F1-measure for the multiclass case first require introducing the concepts of precision, recall, and F1-measure for a particular class $c \in \{1, \dots, C\}$:

$$PREC(c) = \frac{TP(c)}{TP(c) + FP(c)},$$

$$REC(c) = \frac{TP(c)}{TP(c) + FN(c)},$$

and

$$F1(c) = \frac{2PREC(c)REC(c)}{PREC(c) + REC(c)}.$$

Having defined the precision, recall, and F1-measure associated with a particular class $c \in \{1, \dots, C\}$, we can finally define the precision, recall, and F1-measure in the macro, weighted, and micro cases. Specifically, the precision macro, recall macro and F1-measure macro are respectively defined as follows:

$$PREC_MACRO = \frac{1}{C}\sum_{c=1}^{C} PREC(c), \tag{46}$$

$$REC_MACRO = \frac{1}{C}\sum_{c=1}^{C} REC(c), \tag{47}$$

and

$$F1_MACRO = \frac{1}{C}\sum_{c=1}^{C} F1(c). \tag{48}$$

Table 3
Comparisons between the κ-logistic and the classical logistic classifier for binary data sets in terms of mean and standard deviation across the five train/test split.

Data set	Model	ACC	F1	PREC	REC
Heart feailure	κ-logistic	0.82 ± 0.05	0.65 ± 0.05	0.83 ± 0.14	0.63 ± 0.07
	Logistic	0.77 ± 0.04	0.54 ± 0.06	0.71 ± 0.13	0.47 ± 0.06
Hepatitis	κ-logistic	0.81 ± 0.05	0.88 ± 0.03	0.85 ± 0.05	0.92 ± 0.03
	Logistic	0.77 ± 0.02	0.87 ± 0.02	0.81 ± 0.03	0.92 ± 0.04
Mushroom	κ-logistic	0.9 ± 0.01	0.89 ± 0.01	0.96 ± 0.01	0.83 ± 0.02
	Logistic	0.86 ± 0.05	0.85 ± 0.04	0.88 ± 0.1	0.8 ± 0.03
Skin	κ-logistic	0.94 ± 0.0	0.96 ± 0.0	1.0 ± 0.0	0.93 ± 0.0
	Logistic	0.92 ± 0.01	0.95 ± 0.0	0.95 ± 0.01	0.94 ± 0.0

With a weighted average, we account for the contribution of each class $c \in \{1, \dots, C\}$ weighted by the number n_c of instances of that class. With this premise, (46)–(48) respectively become:

$$PREC_WEIGHTED = \sum_{c=1}^{C} \frac{n_c}{K} PREC(c),$$

$$REC_WEIGHTED = \sum_{c=1}^{C} \frac{n_c}{K} REC(c),$$

and

$$F1_WEIGHTED = \sum_{c=1}^{C} \frac{n_c}{K} F1(c),$$

where n_c/K is the ratio of instances of class c over the total number of instances in the test set. Finally, micro precision, recall and F1-measure are respectively defined as

$$PREC_MICRO = \frac{\sum_{c=1}^{C} TP(c)}{\sum_{c=1}^{C} (TP(c) + FP(c))},$$

$$REC_MICRO = \frac{\sum_{c=1}^{C} TP(c)}{\sum_{c=1}^{C} (TP(c) + FN(c))},$$

and

$$F1_MICRO = \frac{2\sum_{c=1}^{C} TP(c)}{\sum_{c=1}^{C} (2TP(c) + FP(c) + FN(c))}.$$

7.3. The models

We compared the classical logistic classifier with the κ-logistic classifier in both binary and multiclass cases. For the basis functions, we chose setting $P = M$ and $\phi_j(\mathbf{x}) = x_j \ \forall\ j \in \{1, \dots, M\}$, $\forall\ \mathbf{x} \in \mathbb{R}^M$. The gradient-based algorithm used for both classifiers has two hyper-parameters: the maximum number of iterations n_{iter} and a positive number a used to reduce the coefficient η_i in the parameter updating formula (including κ). In particular, the index i denotes the ith iteration of the gradient algorithm and η_i decreases according to the following formula:

$$\eta_i = \frac{a}{i+1}.$$

We set these values to $n_{iter} = 1000$ and $a = 1$.

For the κ-logistic classifier, it is important to set a suitable initial value for κ. Given that the κ-loss function is not convex, it is crucial to initialize κ as close as possible to the global minimum to facilitate the use of gradient-based optimization methods. Considering the initial κ value, denoted as κ_0, as a hyperparameter of the model, we selected it using a hyperparameter optimization strategy based on Bayesian Optimization (BO). BO is a sequential model-based approach used to solve complex global optimization problems based on expensive-to-evaluate black-box functions, but also commonly used for tuning the hyper-parameters of machine learning algorithms [56–58]. The selection of κ_0, using BO, is done as follows. Fixing the train-test split, the train set is divided into 5 subsets. For any possible κ_0 suggested by BO, the κ-logistic is trained for 5 times, using 4 folds as a training set, and the 5th set to assess the accuracy of the model. The performance metric function optimized by BO consists of the average accuracy on the 5 different train-test splits. At the end of the hyperparameter optimization process, we obtain a final κ_0 value, that is passed to the $\kappa-$logistic classifier.

Table 4
Comparisons between the κ-logistic and the classical logistic classifier for multiclass data sets in terms of mean and standard deviation across the five train/test split.

Data set	Cirrhosis		Heart disease		Wine		Wireless	
Model	κ-logistic	Logistic	κ-logistic	Logistic	κ-logistic	Logistic	κ-logistic	Logistic
F1_MICRO	0.67 ± 0.07	0.57 ± 0.08	0.61 ± 0.02	0.59 ± 0.01	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
PREC_MICRO	0.67 ± 0.07	0.57 ± 0.08	0.61 ± 0.02	0.59 ± 0.01	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
REC_MICRO	0.67 ± 0.07	0.57 ± 0.08	0.61 ± 0.02	0.59 ± 0.01	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
F1_MACRO	0.44 ± 0.03	0.39 ± 0.08	0.30 ± 0.04	0.27 ± 0.02	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
PREC_MACRO	0.45 ± 0.02	0.38 ± 0.10	0.38 ± 0.07	0.26 ± 0.02	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
REC_MACRO	0.45 ± 0.03	0.40 ± 0.05	0.33 ± 0.04	0.32 ± 0.02	0.98 ± 0.02	0.96 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
F1_WEIGHTED	0.63 ± 0.05	0.55 ± 0.10	0.55 ± 0.02	0.53 ± 0.01	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
PREC_WEIGHTED	0.62 ± 0.04	0.54 ± 0.12	0.56 ± 0.06	0.49 ± 0.02	0.97 ± 0.02	0.95 ± 0.04	0.97 ± 0.00	0.98 ± 0.00
REC_WEIGHTED	0.67 ± 0.07	0.57 ± 0.08	0.61 ± 0.02	0.59 ± 0.01	0.97 ± 0.02	0.94 ± 0.04	0.97 ± 0.00	0.98 ± 0.00

Table 5
Comparisons between κ-logistic, k-NN, and SVM classifiers for binary-class data sets in terms of mean and standard deviation across the five train/test splits.

Data set	Model	ACC	F1	PREC	REC
Heart failure	κ-Logistic	0.82 ± 0.05	0.65 ± 0.05	0.83 ± 0.14	0.63 ± 0.07
	k-NN	0.73 ± 0.03	0.60 ± 0.06	0.64 ± 0.09	0.53 ± 0.11
	SVM	0.78 ± 0.02	0.60 ± 0.04	0.80 ± 0.11	0.47 ± 0.10
Hepatitis	κ-Logistic	0.81 ± 0.05	0.88 ± 0.03	0.88 ± 0.05	0.88 ± 0.03
	k-NN	0.81 ± 0.03	0.88 ± 0.02	0.88 ± 0.01	0.88 ± 0.04
	SVM	0.84 ± 0.02	0.90 ± 0.01	0.88 ± 0.02	0.92 ± 0.03
Mushroom	κ-Logistic	0.90 ± 0.00	0.89 ± 0.00	0.96 ± 0.01	0.84 ± 0.02
	k-NN	0.99 ± 0.00	0.99 ± 0.00	1.0 ± 0.00	0.99 ± 0.00
	SVM	0.96 ± 0.00	0.96 ± 0.00	0.94 ± 0.01	0.98 ± 0.01
Skin	κ-Logistic	0.94 ± 0.00	0.96 ± 0.00	1.0 ± 0.01	0.93 ± 0.00
	k-NN	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00
	SVM	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00	1.0 ± 0.00

7.4. κ−logistic vs. classical logistic

For each data set, we compared the performance of classical and κ-logistic classifiers, obtained by splitting each data set in training (80%) and test set (20%). We repeated the overall procedure five times using a 5-fold cross-validation procedure to evaluate performance variability in terms of mean and standard deviation.

In Table 3, we report the comparison of the performances between the κ-logistic and the classical logistic classifier for the binary data sets. More in detail, for each data set and model, we report the accuracy, precision, recall and F1-measure in terms of mean and standard deviation across the five train/test splits. We can clearly observe that the κ-logistic classifier generally outperforms the classical logistic one.

In Table 4, we report the comparison of the performances between the κ-logistic and the classical logistic classifier for the multiclass data sets. More in detail, for each data set and model, we report the accuracy, F1-measure (micro, macro, and weighted), precision (micro, macro, and weighted) and recall (micro, macro, and weighted) in terms of mean and standard deviation across the five train/test splits. Again, we observe that the κ-logistic classifier outperforms the classical logistic classifier, except for the wireless dataset. However, this difference is minimal.

Taken together, these results confirm the improvement of κ-logistic over logistic classifiers.

7.5. κ−logistic vs. other classifiers

After comparing the κ-logistic classifiers (binary and multiclass) with their classical logistic counterparts, we now aim to extend our comparison to other types of classifiers. In particular, we have carried out a comparison with k-Nearest Neighbor (k-NN) and Support Vector Machine (SVM) classifiers.

In Table 5, we compare the binary κ-logistic classifier with k-NN and SVM on data sets containing only two labels, while in Table 6, the comparison is made between the multiclass κ-logistic classifier and k-NN and SVM on multiclass data sets.

From Table 5, we observe that the binary κ-logistic classifier outperforms both k-NN and SVM on the heart failure data set. As for the hepatitis data set, the performance is identical in terms of precision, whereas in terms of accuracy, SVM offers the best results. Still regarding precision, k-NN achieves the most promising score on the mushroom data set, followed by k-logistic with 0.96, which in turn outperforms SVM. Finally, k-logistic performs worse on the skin data set in terms of accuracy, though the precision remains unchanged.

Table 6
Comparisons between κ-logistic, k-NN, and SVM classifiers for multi- class data sets in terms of mean and standard deviation across the five train/test splits.

Data set	Model	F1_MICRO	PREC_MICRO	REC_MICRO	F1_MACRO	PREC_MACRO	REC_MACRO	F1_WEIGHTED	PREC_WEIGHTED	REC_WEIGHTED
Cirrhosis	k-Logistic	0.67 ± 0.03	0.67 ± 0.03	0.67 ± 0.03	0.44 ± 0.02	0.47 ± 0.02	0.45 ± 0.02	0.64 ± 0.03	0.65 ± 0.03	0.67 ± 0.03
	k-NN	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04	0.44 ± 0.03	0.43 ± 0.03	0.45 ± 0.03	0.63 ± 0.04	0.62 ± 0.03	0.65 ± 0.04
	SVM	0.69 ± 0.03	0.69 ± 0.03	0.69 ± 0.03	0.45 ± 0.03	0.5 ± 0.02	0.46 ± 0.03	0.66 ± 0.04	0.68 ± 0.03	0.69 ± 0.03
Heart disease	k-Logistic	0.60 ± 0.02	0.60 ± 0.02	0.6 ± 0.02	0.32 ± 0.04	0.38 ± 0.07	0.33 ± 0.04	0.56 ± 0.02	0.56 ± 0.06	0.6 ± 0.02
	k-NN	0.57 ± 0.03	0.57 ± 0.03	0.57 ± 0.03	0.29 ± 0.04	0.33 ± 0.04	0.29 ± 0.04	0.54 ± 0.03	0.53 ± 0.04	0.57 ± 0.03
	SVM	0.65 ± 0.03	0.65 ± 0.03	0.65 ± 0.03	0.35 ± 0.05	0.37 ± 0.07	0.35 ± 0.04	0.6 ± 0.05	0.57 ± 0.07	0.65 ± 0.03
Wine	k-Logistic	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.98 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03
	k-NN	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.02	0.97 ± 0.02	0.98 ± 0.02	0.97 ± 0.03	0.97 ± 0.02	0.97 ± 0.03
	SVM	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.98 ± 0.03	0.98 ± 0.03	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03
Wireless	k-Logistic	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00
	k-NN	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.97 ± 0.01	0.98 ± 0.01	0.98 ± 0.01
	SVM	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01

As for the performance on multi-class data sets (please refer to Table 6), the κ-logistic classifier outperforms k-NN on the cirrhosis data set, while the best performance is offered by the SVM classifier. However, the results in terms of F1_MACRO are competitive. On the heart disease data set, SVM dominates in terms of F1_MICRO, whereas the κ-logistic classifier leads in terms of PREC_MACRO. Performance is essentially identical on the remaining data sets, with the average metric values for κ-logistic being one percentage point lower on the wireless data set, but with a virtually null standard deviation.

Two key insights emerge from the results presented in Tables 5 and 6. First, none of the three classifiers consistently outperforms the others across all data sets. This outcome is expected and aligns with the well-established "No Free Lunch" principle in machine learning: no single classifier can universally dominate others regardless of the data set.

Second, on average, the SVM classifier tends to achieve the best performance. This is hardly surprising, given the inherently more sophisticated internal architecture of the SVM compared to k-NN, logistic regression, and our proposed κ-logistic classifiers.

Nevertheless, the fact that the κ-logistic classifiers demonstrate performance comparable to, and in some cases surpassing, SVMs across multiple data sets strongly indicates their competitiveness. These results validate the κ-logistic classifiers we propose here as viable alternatives to existing methods in the literature and in practical applications.

7.6. Stability

An important aspect concerning the gradient-based method is the concept of stability, namely the guarantee that the algorithm does not diverge or oscillate. In this regard, the choice of the learning rate η plays a crucial role: excessively large values of η may cause oscillations, while very small values certainly prevent such instability but may lead to extremely slow convergence. For this reason, η is often not kept fixed, but rather decreased over the course of the iterations. Since η can lead to instability issues, in this section we aim to verify that the introduction of κ, together with its update within the gradient-based procedure, does not give rise to any stability problems either. To this end, we selected two data sets previously used in our computational experiments, one for the binary case and one for the multi-class case. Specifically, we used the heart disease data set for the binary scenario and the wine data set for the multi-class scenario.

Starting from values of κ_0 around 0.5–namely $\kappa_0 \in \{0.498, 0.499, 0.500, 0.501, 0.502\}$–we tracked the evolution of κ and of the loss function E_κ over the iterations for both data sets, this time limiting the numer of iterations to $n_{iter} = 100$. The corresponding plots are reported in Figs. 3–6. These plots reveal several significant findings.

First, in the long run, neither divergence nor oscillation is observed, which already provides strong evidence of the algorithm's stability. However, an even more insightful observation lies in the fact that the slope of the plotted curves becomes essentially zero at convergence. This indicates not only that the algorithm is stable, but also that it is actually converging to a local minimum, characterized by a vanishing gradient.

Finally, it is interesting to note that for both the heart disease and wine data sets, the values of κ converge to symmetric values with respect to the horizontal axis. This phenomenon reflects the aforementioned symmetry property that $\exp_{-\kappa}(x) = \exp_\kappa(x)$ for all $x \in \mathbb{R}$, suggesting that the algorithm effectively converges to a unique value of κ, modulo this symmetry.

8. Conclusions

In this article, we proposed two κ-logistic classification models. These models use the $\exp_\kappa$ function: a function arising in physics and whose applications appear in many other fields. We introduced two models: one for the binary case and one for multiple classes. The validity of our models is confirmed by a number of aspects: (1) when κ approaches 0, we found the same formulas of the corresponding models used in logistic regression, (2) we proved that the computational complexity of our models is the same as that of the corresponding logistic regression models, and (3) the theoretical results have been confirmed by numerous computational experiments conducted on multiple data sets, both in the binary and multiclass cases. Inspired by these promising results, we believe this theoretical work can be the starting point for a new research direction consisting of the application of the $\exp_\kappa$ function to other machine-learning techniques, such as neural networks and deep learning. Furthermore, another research direction could be to compare the κ-logistic models with logistic models regarding the saturation problem [44].

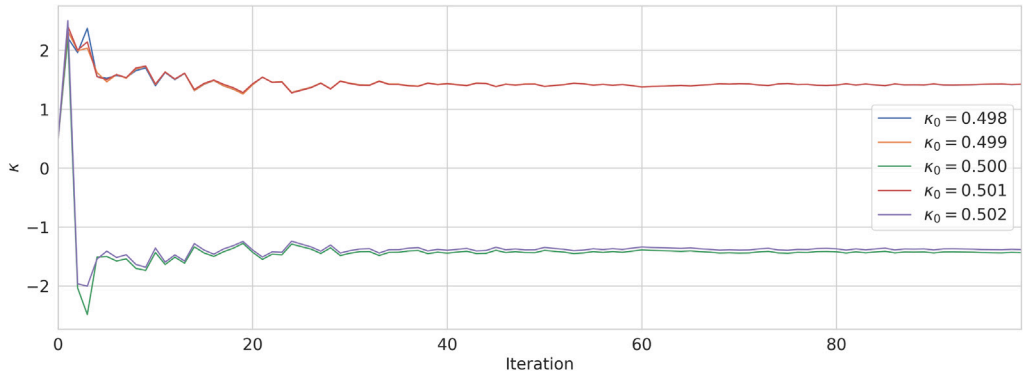

Fig. 3. The evolution of κ for the heart disease data set and various initial values κ_0.

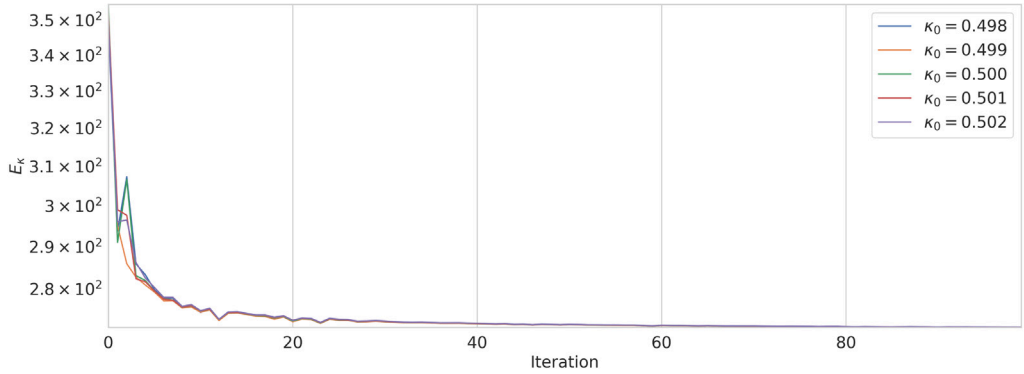

Fig. 4. The evolution of E_κ for the heart disease data set and various initial values κ_0.

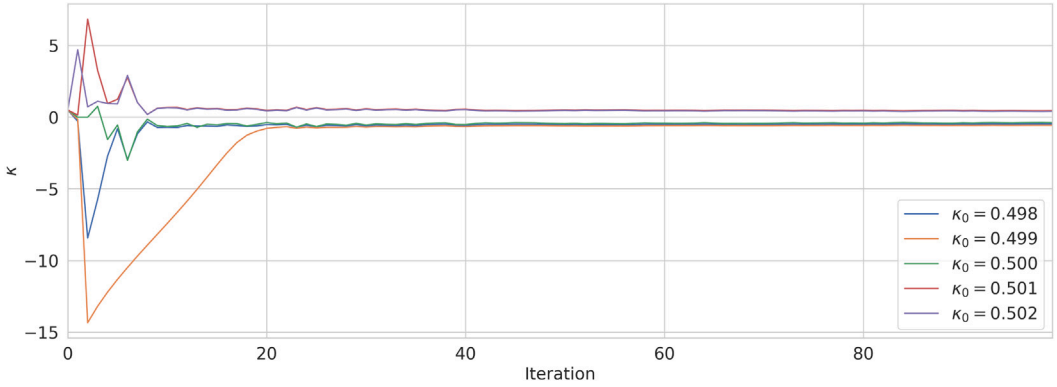

Fig. 5. The evolution of κ for the wine data set and various initial values κ_0.

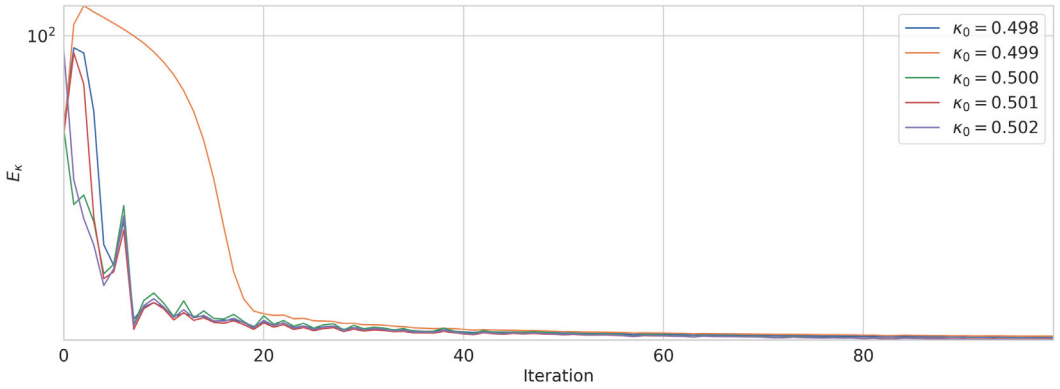

Fig. 6. The evolution of E_κ for the wine data set and various initial values κ_0.

CRediT authorship contribution statement

Mauro Maria Baldi: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Bruno Giovanni Galuzzi:** Writing – review & editing, Validation, Software, Investigation, Formal analysis, Data curation, Conceptualization. **Enza Messina:** Writing – review & editing, Conceptualization. **Giorgio Kaniadakis:** Writing – review & editing, Conceptualization.

Use of generative AI and AI-assisted technologies

During the preparation of this work the authors used chatGPT 3.5 in order to proofread the article. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

Funding

No funding was received for conducting this study.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G. Kaniadakis, Non-linear kinetics underlying generalized statistics, Phys. A 296 (3) (2001) 405–425.
[2] G. Kaniadakis, Statistical mechanics in the context of special relativity, Phys. Rev. E 66 (5) (2002) 056125 1–17.
[3] F. Clementi, M. Gallegati, G. Kaniadakis, A model of personal income distribution with application to Italian data, Empir. Econ. 39 (2010) 559—-591.
[4] A. Vallejos, I. Ormazábal, F.A. Borotto, H.F. Astudillo, A new κ-deformed parametric model for the size distribution of wealth, Phys. A 514 (2019) 819–829.
[5] M.M. Baldi, C. Mammana, E. Michetti, The κ-logistic growth model. Qualitative and quantitative dynamics, Math. Comput. Simul. 225 (2024) 350–369, http://dx.doi.org/10.1016/j.matcom.2024.05.016.
[6] M.M. Baldi, C. Mammana, E. Michetti, Population dynamics and economic growth. Co-evolution and complexity, in: S. Spalletti, F. Spigarelli (Eds.), A Contemporary Comprehensive Economic Science: The Economic Thought and Legacy of Maffeo Pantaleoni, in: Routledge Studies in the History of Economics, Routledge, 2026, ISBN 9781041005032.
[7] B. Trivellato, Deformed exponentials and applications to finance, Entropy 15 (9) (2013) 3471–3489.
[8] O.J. Tapiero, A maximum (non-extensive) entropy approach to equity options bid–ask spread, Phys. A 392 (14) (2013) 3051–3060.
[9] E. Moretto, S. Pasquali, B. Trivellato, A non–Gaussian option pricing model based on Kaniadakis exponential deformation, Eur. Phys. J. B 90 (179) (2017).
[10] B. Lei, J.-I. Fan, Adaptive Kaniadakis entropy thresholding segmentation algorithmbased on particle swarm optimization, Soft Comput. 24 (2020) 7305–7318.
[11] B. Lei, J. Fan, Infrared pedestrian segmentation algorithm based on the two-dimensional Kaniadakis entropy thresholding, Knowl.-Based Syst. 225 (2021) 107089.
[12] L. Rong, G. Xiaoge, Threshold segmentation based on fuzzy Kaniadakis entropy for criminal investigation images, in: AIPR 2021: 4th International Conference on Artificial Intelligence and Pattern Recognition, Xiamen, China, September 24 - 26, 2021, ACM, ISBN: 978-1-4503-8408-7, 2021, pp. 102–108, http://dx.doi.org/10.1145/3488933.3488950.
[13] G. Kaniadakis, M.M. Baldi, Th.S. Deisboeck, G. Grisolia, S.M. Scarfone, A. Sparavigna, U. Lucia, The κ-statistics approach to epidemiology, Sci. Rep. 10 (19949) (2020) 1–14, http://dx.doi.org/10.1038/s41598-020-76673-3.
[14] G. Kaniadakis, New power-law tailed distributions emerging in κ-statistics$^{(a)}$, EPL (Eur. Lett.) 133 (1) (2021) 10002.
[15] G. Kaniadakis, Relativistic roots of κ-entropy, Entropy 26 (5) (2024) 406.

[16] L.A. Passos, M.C. Santana, T. Moreira, J.P. Papa, κ-Entropy based restricted Boltzmann machines, in: 2019 International Joint Conference on Neural Networks, IJCNN, 2019, pp. 1–8.
[17] D. Cox, Analysis of Binary Data, Chapman and Hall, London, 1969.
[18] D. Sankoff, W. Labov, On the uses of variable rules, Lang. Soc. 8 (2–3) (1979) 189–222.
[19] M.A. Richardson, T. Sanders, J.L. Palmer, A. Greisinger, S.E. Singletary, Complementary/alternative medicine use in a comprehensive cancer center and the implications for oncology, J. Clin. Oncol. 18 (13) (2000) 2505–2514.
[20] J. Wang, Q. Zu, W. Wang, Analysis of factors of pulmonary fungal infection in mice in respiratory medicine department based on logistic regression analysis model and Progranulin, Saudi J. Biol. Sci. 27 (2) (2020) 629–635.
[21] T. Vlandas, Grey power and the economy: Aging and inflation across advanced economies, Comp. Political Stud. 51 (4) (2018) 514–552.
[22] X. Liang, X. Hu, J. Jiang, Research on the effects of information description on crowdfunding success within a sustainable economy—The perspective of information communication, Sustainability 12 (2) (2020) 650.
[23] A. Kartal, Balance scorecard application to predict business success with logistic regression, J. Adv. Econ. Financ. 3 (1) (2018) 13.
[24] L. Svabova, L. Michalkova, M. Durica, E. Nica, Business failure prediction for Slovak small and medium-sized companies, Sustainability 12 (11) (2020) 4572.
[25] K. Nigam, J. Lafferty, A. McCallum, Using maximum entropy for text classification, in: IJCAI-99 Workshop on Machine Learning for Information Filtering, Vol. 1, Stockholom, Sweden, 1999, pp. 61–67.
[26] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, ISBN: 978-0387-31073-2, 2006.
[27] Y. Tsuruoka, J. McNaught, J. Tsujii, S. Ananiadou, Learning string similarity measures for gene/protein name dictionary look-up using logistic regression, Bioinformatics 23 (20) (2007) 2768–2774.
[28] M.G. Philiastides, P. Sajda, Temporal characterization of the neural correlates of perceptual decision making in the human brain, Cerebral Cortex 16 (4) (2006) 509–518.
[29] R. Maranzato, A. Pereira, M. Neubert, A.P. do Lago, Fraud detection in reputation systems in E-markets using logistic regression and stepwise optimization, SIGAPP Appl. Comput. Rev. 11 (1) (2010) 14—-26.
[30] A.B. Owen, Infinitely imbalanced logistic regression, J. Mach. Learn. Res. 8 (4) (2007) 761–773.
[31] K. Jaskie, C. Elkan, A. Spanias, A modified logistic regression for positive and unlabeled learning, in: 2019 53rd Asilomar Conference on Signals, Systems, and Computers, 2019, pp. 2007–2011.
[32] J. Shi, W. Yin, S. Osher, P. Sajda, A fast hybrid algorithm for large-scale l1-regularized logistic regression, J. Mach. Learn. Res. 11 (2010) 713–741.
[33] L. Ren, L. Du, L. Carin, D.B. Dunson, Logistic stick-breaking process, J. Mach. Learn. Res. 12 (1) (2011) 203–239.
[34] G. Yuan, C. Ho, C. Lin, An improved glmnet for l1-regularized logistic regression, J. Mach. Learn. Res. 13 (1) (2012) 1999–2030.
[35] Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, Chih-Jen Lin, A comparison of optimization methods and software for large-scale l1-regularized linear classification, J. Mach. Learn. Res. 11 (2010) 3183–3234.
[36] J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, J. Stat. Softw. 33 (1) (2010) 1–22.
[37] R. Wang, N. Xiu, S. Zhou, An extended Newton-type algorithm for l2-regularized sparse logistic regression and its efficiency for classifying large-scale datasets, J. Comput. Appl. Math. 397 (2021) 113656, http://dx.doi.org/10.1016/j.cam.2021.113656.
[38] N.A. Zaidi, G.I. Webb, M.J. Carman, F. Petitjean, J. Cerquides, ALR^n: accelerated higher-order logistic regression, Mach. Learn. 104 (2) (2016) 151–194.
[39] D.J. Foster, S. Kale, H. Luo, M. Mohri, K. Sridharan, Logistic regression: The importance of being improper, in: Conference on Learning Theory, PMLR, 2018, pp. 167–208.
[40] A. Fallah Tehrani, K.W. Weiwei Cheng, E. Hüllermeier, Learning monotone nonlinear models usingthe Choquet integral, Mach. Learn. 89 (1) (2012) 183–211.
[41] Eyke Hüllermeier, Ali Fallah Tehrani, Efficient learning of classifiers based on the 2-additive Choquet integral, in: Christian Moewes, Andreas Nürnberger (Eds.), Computational Intelligence in Intelligent Data Analysis, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN: 978-3-642-32378-2, 2013, pp. 17–29.
[42] Sadegh Abbaszadeh, Eyke Hüllermeier, Machine learning with the sugeno integral: The case of binary classification, IEEE Trans. Fuzzy Syst. 29 (12) (2021) 3723–3733, http://dx.doi.org/10.1109/TFUZZ.2020.3026144.
[43] S.-C. Kim, A.S. Arun, M.E. Ahsen, R. Vogel, G. Stolovitzky, The Fermi–Dirac distribution provides a calibrated probabilistic output for binary classifiers, Proc. Natl. Acad. Sci. 118 (34) (2021) e2100761118.
[44] V.L. Xavier, Hyperbolic regression: A new regression model with applications to the binary classification problem, Inf. Sci. 609 (2022) 15–25.
[45] C.L. Urbano-Leon, A.M. Aguilera, M. Escabias, Repeated measures in functional logistic regression, Math. Comput. Simul. 225 (2024) 66–77, http://dx.doi.org/10.1016/j.matcom.2024.05.002.
[46] S. Raschka, Python Machine Learning, Packt Publishing, 2017.
[47] Uci, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/index.php, n.d.
[48] Heart Failure Clinical Records, UCI Machine Learning Repository, 2020, http://dx.doi.org/10.24432/C5Z89R.
[49] Hepatitis, UCI Machine Learning Repository, 1988, http://dx.doi.org/10.24432/C5Q59J.
[50] Mushroom, UCI Machine Learning Repository, 1987, http://dx.doi.org/10.24432/C5959T.
[51] Rajen Bhatt, Abhinav Dhall, Skin Segmentation, UCI Machine Learning Repository, 2012, http://dx.doi.org/10.24432/C5T30C.
[52] E. Dickson, P. Grambsch, T. Fleming, L. Fisher, A. Langworthy, Cirrhosis Patient Survival Prediction, UCI Machine Learning Repository, 2023, http://dx.doi.org/10.24432/C5R02G.
[53] A. Janosi, W. Steinbrunn, M. Pfisterer, R. Detrano, Heart Disease, UCI Machine Learning Repository, 1988, http://dx.doi.org/10.24432/C52P4X.
[54] S. Aeberhard, M. Forina, Wine, UCI Machine Learning Repository, 1991, http://dx.doi.org/10.24432/C5PC7J.
[55] Rajen Bhatt, Wireless Indoor Localization, UCI Machine Learning Repository, 2017, http://dx.doi.org/10.24432/C51880.
[56] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, Adv. Neural Inf. Process. Syst. 25 (2012).
[57] B.G. Galuzzi, I. Giordani, A. Candelieri, R. Perego, F. Archetti, Hyperparameter optimization for recommender systems through Bayesian optimization, Comput. Manag. Sci. 17 (2020) 495–515.
[58] A.H. Victoria, G. Maragatham, Automatic tuning of hyperparameters using Bayesian optimization, Evol. Syst. 12 (1) (2021) 217–223.