Check for updates

ReliefE: feature ranking in high-dimensional spaces via manifold embeddings

Blaž Škrlj[1,2] · Sašo Džeroski[1,2] · Nada Lavrač[1,3] · Matej Petković[1,2]

Abstract

Feature ranking has been widely adopted in machine learning applications such as high-throughput biology and social sciences. The approaches of the popular Relief family of algorithms assign importances to features by iteratively accounting for nearest relevant and irrelevant instances. Despite their high utility, these algorithms can be computationally expensive and not-well suited for high-dimensional sparse input spaces. In contrast, recent embedding-based methods learn compact, low-dimensional representations, potentially facilitating down-stream learning capabilities of conventional learners. This paper explores how the Relief branch of algorithms can be adapted to benefit from (Riemannian) manifold-based embeddings of instance and target spaces, where a given embedding's dimensionality is intrinsic to the dimensionality of the considered data set. The developed ReliefE algorithm is faster and can result in better feature rankings, as shown by our evaluation on 20 real-life data sets for multi-class and multi-label classification tasks. The utility of ReliefE for high-dimensional data sets is ensured by its implementation that utilizes sparse matrix algebraic operations. Finally, the relation of ReliefE to other ranking algorithms is studied via the Fuzzy Jaccard Index.

Keywords Feature ranking · Representation learning · Relief

✉ Matej Petković
matej.petkovic@ijs.si

Blaž Škrlj
blaz.skrlj@ijs.si

Sašo Džeroski
saso.dzeroski@ijs.si

Nada Lavrač
nada.lavrac@ijs.si

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

[2] Jožef Stefan International Postgraduate School, Jamova 39, 1000 Ljubljana, Slovenia

[3] University of Nova Gorica, Vipavska 13, 5000 Nova Gorica, Slovenia

1 Introduction

Contemporary machine learning has found its use in many scientific disciplines, ranging from biology, sociology, logistics and engineering sciences to physics. Data sets are often available in tabular form and consist of instances (rows) and features (columns), where attributes denote column names and individual features correspond to individual attribute values.

Even though predictive models can offer insights into how well a certain aspect of a given system can be predicted, researchers and industry practitioners are frequently interested in *which* parts of the input space are the most relevant. Having such knowledge can yield novel insights into relevant aspects of the studied problem, leading to improved human understanding of the studied phenomenon. For example, in modern molecular and systems biology, discovery of novel biomarkers is of high relevance— once the researchers know which, e.g., compounds or proteins indicate the presence of the studied condition, they can be used for preliminary condition detection, but also to advance the human understanding of the conditions leading to the construction of novel hypotheses. We next discuss the types of feature ranking algorithms.

Feature ranking algorithms can be split into two main groups: myopic and non-myopic. Myopic algorithms do not consider multiple features simultaneously and thus potentially neglect *interactions* between features. Examples of myopic feature ranking algorithms include, e.g., the information gain-based ranking. Algorithms from the non-myopic Relief branch, originating in the early Relief algorithm (Kira and Rendell 1992), are among the most widely used non-myopic algorithms for *feature ranking*, where each feature is assigned a real-valued score, offering insights into its importance. The Relief family of algorithms has been successfully applied to numerous real-life problems (Stiglic and Kokol 2010; Stokes and Visweswaran 2012; Petković et al. 2021). In this work we propose ReliefE, an embedding-based feature ranking algorithm built on the ideas of the original Relief and ReliefF (Robnik-Šikonja and Kononenko 2003), as well as their extensions to a multi-label classification setting (Petković et al. 2018). ReliefE does not compute feature importances based on the original, high-dimensional feature space, but via low-dimensional embeddings of the input and/or output spaces. The key contributions of this work are summarized as follows:

- We present ReliefE, an algorithm for feature ranking implemented using sparse matrix algebraic computation of distances between low-dimensional manifold embeddings of both instances and targets (when considering multi-label classification).
- The latent dimension of the space, in which the distances are computed, is *inferred automatically* in an efficient manner.
- We show that the number of neighbors to be considered can be automatically inferred based on the distribution of distances to the considered instances, rather than hard-coded as part of the input.
- Theoretically grounded sparsification of the input was considered as a preprocessing step, potentially decreasing the execution time.
- We offer evidence that ReliefE performs significantly faster than many state-of-the-art methods, especially in high-dimensional input (and output) spaces.
- Theoretical properties of ReliefE, including the properties of the embedding spaces, their relations and computational complexity analysis are studied.

- We showcase the ReliefE's performance against six strong (widely used) baselines on 20 real-life multi-class and multi-label classification data sets.
- We perform extensive Bayesian and frequentist performance comparisons assessing the statistical evidence of ReliefE's utility and potential drawbacks.

The rest of this paper is structured as follows. In Sect. 2, we discuss the key ideas that have led to the developments described in this paper. Section 3 presents the proposed ReliefE methodology, followed by a description of the experimental setting in Sect. 4 and the results of the empirical evaluation in Sect. 5. The overall findings are discussed in Sect. 6, followed by the conclusions and plans for further work in Sect. 7. The paper includes numerous appendices presenting detailed results of empirical comparisons and case studies.

2 Background

In this section we discuss the works that have impacted this paper the most, starting with the notion of feature ranking and the description of the Relief branch of feature ranking algorithms. Next, we discuss how embedding-based learning can be of use when solving otherwise intractable problems, serving as a motivation for the proposed work.

2.1 Feature ranking

Feature ranking can be considered as the process of learning to prioritize the feature space with respect to a given learning task. Algorithms that rank features can operate in non-myopic (considering feature interactions) or myopic manner (ignoring feature interactions). One of the first and most widely used algorithms for non-myopic feature ranking is Relief (Kira and Rendell 1992), introduced in the early 1990s. This iterative algorithm operates by randomly selecting parts of the instance space (e.g., rows), followed by iterative update of weights corresponding to individual features based on the closest instances from the positive and negative classes. The original Relief performs well for binary classification, however was unable to generalize to more complex learning tasks such as multi-class classification. Its extension, ReliefF (Robnik-Šikonja and Kononenko 2003), introduced a prior-based weighting scheme that can take different classes into account. In the following years, multiple adaptations of both Relief and ReliefF were introduced, varying mostly in terms of schemes for taking into account a given instance's neighborhood and its (aggregated) properties. For example, SURF (Greene et al. 2009) unifies the considered per-class neighborhoods, whereas SURFstar (Greene et al. 2010) additionally considers distant neighbors. Further, MultiSURFstar (Granizo-Mackenzie and Moore 2013) takes neighborhood boundaries into account based on the average and standard deviation of distances from the target instance to all others. The TuRF adaptation (Moore and White 2007) of any Relief algorithm also employs recursive feature elimination, whilst applying the dedicated Relief implementation iteratively during feature pruning. TuRF attempts to address some of the problems that arise in very large feature spaces (e.g., more than 20,000 features), yet at the cost of higher computational complexity. In terms of scalability, for example, VLSReliefF (Eppstein and Haake 2008) samples *random subspaces* whilst simultaneously offering competitive performance at a far lower computational cost. The above

ReliefF variants mostly attempt to correct some of the original ReliefF drawbacks by re-considering the update step and how the neighbors are taken into account.

In recent years, the Relief algorithms have also been extended to a multi-label classification setting (MLC)—a learning problem where multiple labels for an instance are simultaneously possible. Examples of this task include gene function prediction. The first attempt of extending ReliefF to MLC that scales well with the number of labels (Petković et al. 2018) uses the Hamming distance as the distance measure between two label sets. Since Hamming loss can be expressed as a sum of component-wise differences, it is not able to detect the possible label-label interactions, which may lead to sub-optimal quality of the obtained rankings as shown recently (Petković et al. 2018), where other MLC error measure-based distances between labels were shown to offer superior performance.

2.2 Embeddings

The rise of embedding-based learning can be nowadays observed in virtually all areas of science and industry. Since the 2010s, for example, deep neural networks are successfully used in fields such as computer vision, where state-of-the-art results are consistently demonstrated, e.g., in face recognition and anomaly detection (LeCun et al. 2015; Pouyanfar et al. 2018). Further, in the recent years, natural language processing has been transformed first by the introduction of recurrent neural networks, followed by the attention-based neural networks (transformers), showing prominent results in the areas of question answering, language understanding and text classification (Vaswani et al. 2017). Similar results have been observed in the areas of network mining (Kipf and Welling 2016) and time series analysis (based on the initial work of Connor et al. 1994).

Neural networks offer an elegant alternative for learning a latent representation of the input data set that can be *transferred*, as well as directly used for problem solving. More recent works on representation learning, however, suggest that a low-dimensional manifold is a suitable topological object for learning rich and transferable representations (Bronstein et al. 2017; Masci et al. 2015). Even though representations learned by neural networks can be associated with manifold learning, the development of algorithms capable of approximating such low-dimensional manifolds has been an active research area even before the era of deep learning, and is of particular relevance to this work. For example, algorithms such as Isomap (Balasubramanian and Schwartz 2002) and Locally linear embedding (Roweis and Saul 2000) have been successfully used for data visualization and more efficient learning. Further, modern *omics* sciences have widely adopted t-SNE (Maaten and Hinton 2008), an approximation method capable of producing two dimensional embeddings of high-dimensional spaces, making it suitable for e.g., gene expression visualization. Hyperbolic embeddings have been also recently demonstrated to be useful when considering hierarchical multi-label classification (Stepišnik and Kocev 2020).

This work builds on the notion of uniform manifold projections (UMAP) (McInnes et al. 2018a, b), a recently introduced algorithm with a strong theoretical grounding in *manifold theory*. We explore, whether low-dimensional manifold approximations of sparse input spaces can be a natural extension to the Relief family of algorithms.
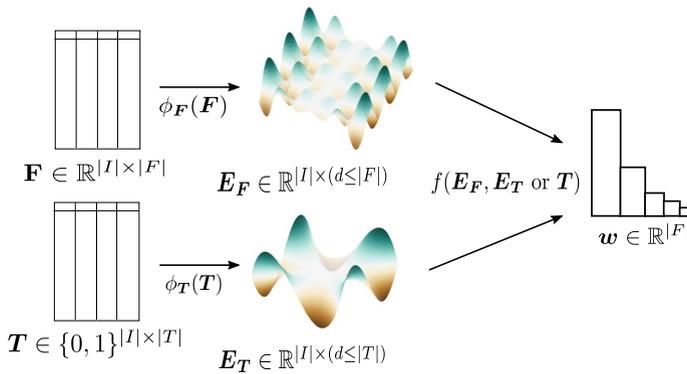
Fig. 1 Overview of the core idea behind ReliefE. Distances in both the feature and target space are computed based on instance embeddings

3 Proposed methodology

One of the main criticisms of the Relief branch of algorithms is their lack of ability to handle very high-dimensional, potentially sparse input spaces, where problems arise either due to increased space complexity or due too incremental weight update steps that result in similar importance scores for many features (i.e. non-informative rankings). We developed the proposed ReliefE algorithm with the goal to addresses these issues. In this section, we discuss in detail ReliefE, the proposed embedding-based feature ranking algorithm for multi-class and multi-label classification problems, along with its implementation, currently one of the fastest Python-based implementations compiled to machine code capable of handling both multi-class and multi-label classification problems.

The proposed ReliefE algorithm is summarized in Fig. 1. Here, the input feature space (F) is mapped (by ϕ_F) to its corresponding low-dimensional approximation (E_F). Finally, the Relief feature ranking (f) is adapted to operate via this low-dimensional representation to obtain final feature rankings w. Further, in a multi-label setting, distances between targets (T) can also be computed in the latent space E_T, constructed via ϕ_T.

3.1 Rationale for embedding-based ranking

Embedding the instances in the feature space prior to feature ranking is sensible due to the fact that volume increases *exponentially* with dimension. Many classifiers benefit from increasing the number of dimensions, however, once a certain dimensionality is reached, their performance can degrade (Hughes 1968). Feature ranking aids this problem by prioritizing parts of the feature space that are relevant for learning.

Higher-dimensional feature spaces render feature importance detection in the initial feature space harder, as more subtle differences between instances need to be taken into account. Embedding (in an unsupervised manner) offers the construction of instance representations that potentially carry additional *semantic context*, as comparing instances in the embedded space does not compare only the considered pairs of instances but also their potential *roles* in the joint latent space, offering more meaningful comparisons (as shown

in e.g., Mikolov et al. 2013 for words and phrases). We next discuss the embedding method considered throughout this work.

3.2 Uniform manifold approximation and projection

This work builds on the recently introduced idea of Uniform Manifold Approximation and Projection (UMAP) (McInnes et al. 2018a) for the task of low-dimensional, unsupervised representation learning. Even though a detailed treatment of the theoretical underpinnings of UMAP is beyond the scope of this paper, we discuss some of the key ideas underlying the actual implementation, and refer the interested reader to McInnes et al. (2018a) for a detailed overview of the theory.

UMAP is formulated with concepts from both topological manifold theory and applied category theory. Riemannian manifolds are topological spaces that have a locally constant metric, and are locally connected. UMAP assumes that high dimensional data can be uniformly mapped across a low-dimensional Riemannian manifold. The locality of the metric, connectivity and uniformity of the mapping are the three main assumptions of this method. Even though such assumptions are not necessarily fulfilled, appropriate selection of the metric used by UMAP can offer better performance and generalization when the learned representations are used for down-stream learning tasks. In contrast to t-SNE, which is effective for two dimensional embeddings, UMAP is highly efficient also for embeddings into *higher-dimensional* vector spaces. UMAP thus serves as a general unsupervised representation learner.[1] It has been successfully used for exploration of biological and other high-dimensional data sets (Cao et al. 2019). In summary, the topological manifold theory underlying UMAP offers a very general representation learning framework, applicable beyond the current implementation of UMAP, which we discuss in more detail below.

Even though the original formulation assumes continuity, in practice, discrete graph-theoretical analogs of the continuous concepts are employed. The representations are a result of a two-step procedure, where in the first step, a weighted graph is constructed based on the distances between the closest instances. The second step resembles conventional graph layout computation, which is normally computed in two or three dimensions for visualization purposes, where, e.g., two coordinates of a 2D layout represent two features. Analogously, UMAP extends the idea to higher dimensions, where the instances are positioned in a d-dimensional space (with d going up to several hundred in most cases). The resulting space is not useful for direct visualization, but serves as a representation suitable for a down-stream learning task—in this work, feature ranking. The computation of the UMAP embedding can be described as follows.

Weighted graph construction. Assume a user-specified dissimilarity measure $\delta : \mathbb{R}^{|F|} \times \mathbb{R}^{|F|} \rightarrow [0, \infty)$ and the number of nearest neighbours k, computed via an approximation algorithm (Dong et al. 2011). We refer to the ordered set of instances nearest to instance r_i as $\{r_{i_1}, \ldots, r_{i_k}\}$. For each instance, let

$$\omega_i = \min \{\delta(r_i, r_{i_j}) | 1 \leq j \leq k, \delta(r_i, r_{i_j}) > 0\},$$

---

[1] UMAP can also perform supervised embeddings, yet this functionality is not considered in this work.

and β_i such that

$$\sum_{j=1}^{k} \exp\left(\frac{-\max(0, \delta(r_i, r_{r_j}) - \omega_i)}{\beta_i}\right) = \log_2(k).$$

Next, a weighted directed graph $G = (N, E, w)$ is constructed, where N is the set of considered instances, $E = \{(r_i, r_{i_j}) | 1 \leq j \leq k\}$ and

$$w((r_i, r_{i_j})) = \exp\left(\frac{-\max(0, \delta(r_i, r_{i_j}) - \omega_i)}{\beta_i}\right).$$

The final adjacency matrix $\boldsymbol{B}$ is computed as $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{A}^T - \boldsymbol{A} \odot \boldsymbol{A}^T$ where, $\boldsymbol{A}$ is the adjacency matrix of G and $\odot$ denotes the Hadamard product.

Layout computation.   In the second step, the graph undergoes a process resembling energy minimization, where repulsive and attractive forces are iteratively applied across pairs of instances, resulting in a d-dimensional layout, which is effectively a real-valued embedding. The attractive forces (F_+) are computed, for a given pair of vertices n_i and n_j and their corresponding coordinate representations (embeddings) $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ as follows:

$$F_+ = \frac{-2 \cdot a \cdot b \cdot ||\boldsymbol{r_i} - \boldsymbol{r_j}||_2^{2(b-1)}}{1 + ||\boldsymbol{r_i} - \boldsymbol{r_j}||_2^2} \cdot w(n_i, n_j) \cdot (\boldsymbol{r_i} - \boldsymbol{r_j}),$$

and similarly, the repulsive forces (F_-) are computed as

$$F_- = \frac{b \cdot (1 - w(n_i, n_j)) \cdot (\boldsymbol{r_i} - \boldsymbol{r_j})}{(\eta + ||\boldsymbol{r_i} - \boldsymbol{r_j}||_2^2)(1 + ||\boldsymbol{r_i} - \boldsymbol{r_j}||_2^2)}.$$

Here, η is introduced for numerical stability (a small constant), while a and b are hyperparameters. Note that the F_- update is computationally very expensive, and is addressed via sampling. The initial coordinates are computed by using spectral layout—here, the two eigenvectors corresponding to the two largest eigenvalues are used as the starting set of coordinates.

The two steps, when implemented efficiently, offer fast construction of d-dimensional, real-valued representations. The considered UMAP implementation exploits the Numba framework (Lam et al. 2015) for compiling parts of Python code, making it scalable whilst maintaining user-friendly API-based execution. Note that UMAP's computational complexity depends heavily on the approximate k-nearest neighbor computation. In the following sections, we discuss how we further facilitate the embedding computation, as the current version of UMAP still has *high space complexity* (graph construction).

3.3 Input sparsification

The proposed methodology is capable of handling highly sparse inputs without additional memory overheads. However, real-life data frequently comes in the form of dense matrices, as is the case with, e.g., gene expression data sets. As a part of the proposed methodology, we explored whether input sparsification can speed up the feature ranking process

with minimal ranking quality degradation. We implemented the theoretically grounded Probabilistic Matrix Sparsification algorithm (PrMS) for matrix sparsification (Arora et al. 2006), given in Algorithm 1.

The mathematical intuition behind PrMS is as follows. Given a real-valued matrix $A \in \mathbb{R}^{n \times n}$, let $S = \sum_{ij} |A_{ij}|$. A single PrMS pass through A guarantees at least $\mathcal{O}(\frac{\sqrt{n \cdot S}}{\Delta})$ elements with probability $1 - \exp(-\Omega(\frac{\sqrt{n \cdot S}}{\Delta}))$. Here, Ω represents the lower bound, and $\Delta = \epsilon / \|A\|_2$, where parameter $\epsilon > 0$ determines the approximation level. Further, with probability $1 - \exp(-\Omega(n))$, $||A - \hat{A}||_2 \leq \mathcal{O}(\epsilon)$ holds.[2] Note that, as the majority of real-life data sets are not represented by symmetric matrices (typically, they are not even square matrices), the transformation $B \mapsto A = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$ of the initial matrix $B \in \mathbb{R}^{m \times n}$ has to be employed since A is symmetric and $\|A\|_2 = \|B\|_2$. We heuristically consider $\epsilon = \|A\|_\infty / (m + n)$, i.e. the maximal column-average (of absolute values) of matrix A.

The sparsification procedure remains highly dependent only on ϵ, the parameter determining the reconstruction error that is allowed.

We have used this estimate of ϵ as it is fast to compute and avoids the need for user specification of ϵ, whilst simultaneously guaranteeing reasonable performance (see Sect. 5). One of the most crucial hyperparameters related to representation learning in general is the dimension of the space in which the constructed representation resides. We have attempted to automate the choice of this—otherwise hard-coded—parameter and discuss the considered estimate next.

3.4 Estimation of latent dimension

Following (Facco et al. 2017), we improve upon the idea of latent dimension estimation via top two nearest neighbors. To compute the latent dimension d of the data (under the assumption of a locally constant probability density), it suffices to define two (hyper) spheres S_1 and S_2. Both are centred at a random data sample and have radii equal to the distances between the sample and its two nearest neighbors (Facco et al. 2017). The radii and the dimension d define the volumes of the spheres and it turns out that the value of d can be easily estimated from the empirical probability distribution of the ratio $V(S_2 \backslash S_1)/V(S_1)$ of the volumes of the shell $S_2 \backslash S_1$ and sphere S_1. The method, implemented as a part of ReliefE is summarized in Algorithm 2.

Algorithm 1: Probabilistic Matrix Sparsification (PrMS) [5]

Data: Input matrix A, number of elements n, approximation constant ϵ

1   **for** $i, j \in n$ **do**
2   **if** $|A_{ij}| > \frac{\epsilon}{\sqrt{n}}$ **then**
3         $\hat{A_{ij}} = A_{ij}$
4   **else**
5         $\hat{A}_{ij} = \begin{cases} \text{sgn}(A_{ij}) \cdot \frac{\epsilon}{\sqrt{n}}; & \text{with probability } p_{ij} = \frac{\sqrt{n}|A_{ij}|}{\epsilon} \\ 0; else \end{cases}$
6   **return** $\hat{A}$

---

[2] For extensive theoretical treatise, please consult (Arora et al. 2006).

Algorithm 2: Latent dimension estimation [16].

Data: Set of instance indices I, (sparse) feature matrix $\mathbf{F} \in \mathbb{R}^{|I| \times |F|}$
1 distanceMatrix $\leftarrow$ pairwiseDistancesNonzero($\boldsymbol{F}$)
2 $\mu \leftarrow$ distanceMatrix[:,1]/ distanceMatrix[:,0]
3 empiricalDist $\leftarrow$ computeEmpiricalDistribution(μ)
4 $d \leftarrow$ leastSquaresLine(log(μ), log(1 - empiricalDist))
5 **return** d

The algorithm first computes distanceMatrix, i.e. a matrix comprised of distances to the top two nearest neighbors. In this work, we improve upon the original idea of simply computing the two nearest neighbors of a given instance. Instead, we ignore the neighbors that are equal to a given instance (are at distance 0), and take into account the two nearest neighbors that are at a positive distance to the given instance (method pairwiseDistancesNonzero). The rationale for this step is that this method is also used on the output space of multi-label classification data, where two (or more) examples can often have the same output value (vector describing the labels assigned to the instance). If we had followed the original method, some components of the vector μ would equal ∞ or NaN. Using the modified procedure, numerical instability is avoided whilst observing the same, or very similar, results. Next, μ, a quotient between the two distance vectors is computed (second closest against closest). An empirical distribution is derived from μ. This distribution can be defined as:

$$\text{EMP}_n(x) = \frac{1}{|I|} \sum \mathbb{1}_{x_i < x},$$

where $\mathbb{1}$ represents the indicator function (presence). Thus, for a given x, F_n represents the relative number of elements that are smaller than x. The logarithms of μ, as well as (1 - $\text{EMP}_n(x)$) are computed next. The line between the two quantities intersects 0, and its coefficient, when rounded to the nearest integer, corresponds to the estimated latent dimension. For example, the intrinsic dimension of the *genes* data set is estimated to 33 (see Fig. 2).

3.5 Scaling up UMAP: learning to embed based on representative subspaces

As the most apparent memory (space) bottleneck, we (empirically) identified the UMAP's graph construction phase, where the entirety of the instance space is used for learning to approximate the low-dimensional manifold (embedding). This paper explores whether representative subspaces of the instance space can serve similarly well for learning to embed. The idea was inspired by recent work on random output space selections for ensemble learning (Breskvar et al. 2018). The two adaptations we needed to consider were the initialization and training on a representative subspace of the instances. In the original UMAP, the initialization is based on the spectral decomposition of the instance graph. We instead considered random initialization, which already notably reduced the memory requirement, and kept the quality of the ranking at the approximately same level. However, on larger data sets (see Sect. 4), the embedding computation was still beyond the capabilities of an off-the-shelf computer (Lenovo Carbon X1). To overcome this issue, we employ the following two-step sampling scheme. In the first step, the set of target values appearing in the data was ordered into a list. In the second step, we *cyclically* iterate through the list and
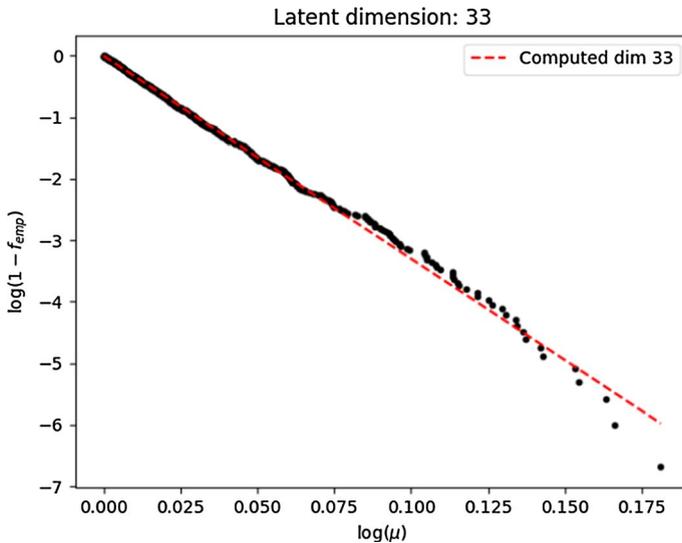
Fig. 2 Intrinsic dimension for the *genes* (Weinstein et al. 2013) data set

(without repetition) choose an element with a given target value, or skip this value if all samples labeled with the considered target value have already been chosen.

The multi-class (and binary) classification examples thus adhere to straightforward mapping from possible classes to the corresponding *multisets* of instances. We also extended this idea to the task of multi-label classification (MLC) where multiple labels can be simultaneously assigned a given example. However, a finite number of possible different label sets allows for first enumerating them, and then proceeding as in the multi-class classification scenario. We consider the theoretical properties of this procedure in Sect. 3.8.

3.6 Adaptive neighbor selection and comparison to average neighbor representation

The first improvement we introduce next, removes the hyperparameter k—the number of neighbors considered for an update step w.r.t. a single randomly chosen instance. Commonly, k is a user defined hyperparameter: However, we explored whether this part of the user input can be removed entirely and replaced by a distance distribution-based heuristic that *dynamically allocates* a number of neighbors suitable for a given randomly selected instance, albeit at some additional computational cost. The rationale for such a heuristic is that real-life data sets are often not uniformly distributed (Liu et al. 2005), indicating only a few other instances are mostly well connected with a given one (scale-free property).

We implement this (optional) estimation as follows. For each instance, ReliefF computes its distance to the remainder of the other instances in order to obtain the top k nearest ones. Such hard-coded selection scheme is not optimal, as it does not take into account the shape of the distance distribution with respect to an individual instance. To overcome this issue, we propose the following procedure:

1. Sort distances w.r.t. the selected instance r_i.

2. Compute the difference between each pair of consequent entries in the distance vector.
3. Select the value of k based on the maximum difference observed.

This procedure intuitively takes into account the shape of the distance distribution as follows. Assuming that all instances are similarly far from the selected instance, the difference vector will mostly consist of small values (a given pair of distances is not all that different). This can result in large k, as the global difference maximum can occur very late. On the contrary, if only a handful of instances are close, and the remainder is very far, only this handful shall serve to determine k, which will be consequently lower. Furthermore, if very high k is selected and all distances are approximately the same, their mean will be similar no matter how many are selected. If a similar situation is considered for highly non-uniform distance distribution, the mean of selected k nearest instances should represent only the ones that are indeed similar to the selected instance, and do not take into account the remainder which is further and possibly not as relevant.

Even if the standard IID assumption holds when sampling a data set, we are always given a finite sample. The neighborhoods of the samples on the border of the convex hull of the data set are most probably different than the neighborhoods of those in the interior of the hull. Additional theoretical properties of this neighbor number selection are given in Sect. 3.8, where computational complexity is also considered.

The proposed adaptation of ReliefF was further extended with an additional option to use the *average neighbor* for the update step, instead of performing updates based on all neighbors and averaging the updates. Albeit subtle, this difference potentially improves the update part's robustness, making the update step less prone to possible outliers amongst the nearest neighbors—such situations could emerge, if the value of k is hard-coded, as is commonly the practice. The intuition behind this incremental change in weight update is as follows. As the distance computation is conducted in the *latent* space, averaging the representation of the neighborhood can also be considered as constructing a *semantic representation* of the considered instance's neighborhood.

3.7 ReliefE—ranking via manifold embeddings

In the following sections, we discuss more formally the proposed ReliefE algorithm incorporating the possible improvements stated in the previous sections. The solution (that handles both multi-class and multi-label classification) is given as Algorithm 3. We can see that this is an iterative algorithm where a random sample r is chosen on each of the n iterations, and distances between r and the remaining instances are computed (lines 6 and 18). We next discuss the two main parts of ReliefE with respect to the addressed classification task.

Algorithm 3: ReliefE.

Data: feature set F, instance index set I, (sparse) feature matrix $\mathbf{F} \in \mathbb{R}^{|I| \times |F|}$, target classes C, (sparse) target space $\mathbf{T} \in \{0, 1\}^{|I| \times |C|}$, neigborhood size k, boolean adaptiveThreshold, distance score δ, latent dimension d, boolean estimateLatentDimension, MLC distance τ, number of iterations s, indices ν of dimension estimation examples

1 $\boldsymbol{w} \leftarrow$ zero list of length $|F|$ ▷ Initiate importances.
2 **if** *estimateLatentDimension* **then**
3 | $d \leftarrow$ latentDimension($\mathbf{F}, \nu$) $(\in \mathbb{N}_+)$
4 $\mathbf{E} \leftarrow$ manifoldProjection($|\mathbf{F}|, d, \nu$) $(\in \mathbb{R}^{|I| \times d})$
5 **for** *i in 1 … s* **do**
6 | $\boldsymbol{r}_i \leftarrow$ randomInstance($\mathbf{F}$)   ▷ Sample instance.
7 | **if** *MCC* **then**
8 | | **for** $c \in C$ **do**
9 | | | dists $\leftarrow$ computeDistances($\boldsymbol{r}_i, \delta, \mathbf{E}, c$)
10 | | | sortedIndices $\leftarrow$ argSortDistances(dists)
11 | | | **if** *adaptiveThreshold* **then**
12 | | | | $k \leftarrow$ adaptiveThresholdMethod(sortedIndices, dists)
13 | | | nearestNeighbors $\leftarrow$ select(sortedIndices, k)
14 | | | **for** j *in* $1 \dots |F|$ **do**
15 | | | | priorWeight $\leftarrow$ ComputeWeight(c, dists, k)
16 | | | | $\boldsymbol{w}[j] \leftarrow \boldsymbol{w}[j] +$ updateScore(priorWeight, nearestNeighbors, j)
17 | **else if** *MLC* **then**
18 | | dists $\leftarrow$ computeDistances($\boldsymbol{r}_i, \delta, \mathbf{E}$)
19 | | sortedIndices $\leftarrow$ argSortDistances(dists)
20 | | **if** *adaptiveThreshold* **then**
21 | | | $k \leftarrow$ adaptiveThresholdMethod(sortedIndices, dists)
22 | | nearestNeighbors $\leftarrow$ sortedIndices[:k]
23 | | targetDistances $\leftarrow \{\}$ ▷ Note the direct use of T compared to MCC.
24 | | descriptiveDistances $\leftarrow \{\}$
25 | | **for** *neighborIndex* $\in$ *nearestNeighbors* **do**
26 | | | distTar $\leftarrow$ distanceToTarget($\boldsymbol{T}[i], \boldsymbol{T}[\text{neighbor}], \tau$)
27 | | | distDes $\leftarrow$ distanceToDesc($\boldsymbol{E}[i], \boldsymbol{E}[\text{neighbor}], \delta$)
28 | | | targetDistances.add(distTar)
29 | | | descriptiveDistances.add(distDes)
30 | | meanTarDist $\leftarrow$ mean(targetDistances)
31 | | meanDesDist $\leftarrow$ mean(descriptiveDistances)
32 | | TD-DIFF, D-DIFF, T-DIFF $\leftarrow$ expectedDist(meanTarDist, meanDesDist)
33 | | **for** j *in* $1 \dots |F|$ **do**
34 | | | $\boldsymbol{w} \leftarrow \boldsymbol{w} + \frac{\text{TD-DIFF}}{\text{T-DIFF}} - \frac{\text{D-DIFF} - \text{TD-DIFF}}{1 - \text{T-DIFF}}$.   ▷ See Eq. 1.
35 **return** $\boldsymbol{w}$

The adaptive threshold step can be further formalized as shown in Algorithm 4.

Algorithm 4: adaptiveThresholdMethod.

Data: distances, sortedIndices
1 sortedDist $\leftarrow$ reorderAscending(dists, sortedIndices)
2 sortedDiff[i]$\leftarrow$ sortedDist$[i+1]-$sortedDist$[i]$ $\left(\in \mathbb{R}_+^{|F|-1}\right)$
3 $k \leftarrow \arg\max$ sortedDiff   ▷ Peak point.
4 **return** k

3.7.1 Multi-class classification

We first discuss the part of ReliefE algorithm that handles multi-class classification (MCC) tasks. Here, the classes are traversed (line 8) as follows. If *adaptiveThreshold* is enabled, the number of neighbors to be considered is determined dynamically for each sample (see Sect. 3.6 for more details). Next, the neighbors are selected and used for the final weight update, where the prior probabilities of individual classes are also considered. The *updateScore* method iteratively updates the weights (line 16), and is in this work for the *j*-th feature and *i*-th sample defined as follows:

$$w[j]+=\underbrace{\left|r_i^j - \mathbb{E}[\text{nearestNeighbors}(i)][j]\right|}_{\text{absMean weight update}} \cdot \underbrace{\begin{cases} P[c]/(1-P[c_i]) \; ; \; c \neq c_i \\ -1 \qquad\qquad\quad ; \; c = c_i \end{cases}}_{\text{Prior information}}.$$

In the proposed update step, an instance is compared directly to the mean of its neighbors which reduces the noise compared to the original updates of Relief (Kira and Rendell 1992) where the order of the averaging and $|\cdot|$ operators is reversed.

The nearestNeighbors represents the ordered set of indices of the top *k* neighbors, thus $\mathbb{E}[\text{nearestNeighbors}(i)][j]$ represents the first moment w.r.t. the *j*-th feature based on the nearest neighbors of the *i*-th instance (there are *k* such neighbors). We set the prior to -1 and the offset for considering the nearest neighbors to +1 if $c = c_i$, i.e., the considered class c_i is equal to the currently considered class *c*.

3.7.2 Multi-label classification

For multi-label classification (MLC option), the weight update step differs substantially from the MCC case, after selecting a random instance (r_i) and determining its distance to the other examples. First, the indices of the closest *k* neighbors are stored in nearest-Neighbors. As the values in the target space T are sets of (multiple) labels per instance, the simple iteration considered in the MCC case does not take the interactions between classes into account (label co-occurrences). Hence, distances are also computed between the target values of r_i and its nearest neighbors (line 26), by using one of the implemented options of ReliefE, which are given in Table 1.

Note that we also consider the cosine and hyperbolic distances which are applicable if the label space *is embedded* prior to the ranking step. We believe employment of manifold projections that operate on sparse spaces can be of relevance for high-dimensional output spaces, as for example seen when considering gene function prediction (Urbanowicz et al. 2018). Once the distances are obtained both based on the input space and the output space,

Table 1 Considered distances between rows in the multi-label output matrix T

Distance	Definition
F1	$\text{dist} = \begin{cases} 1 - \frac{2t_1 t_2^T}{(\text{nnz}(t_1) + \text{nnz}(t_2))} & ; (\text{nnz}(t_1) + \text{nnz}(t_2)) > 0 \\ 0 & ; \text{otherwise} \end{cases}$
Accuracy	$\text{dist} = \begin{cases} 1 - \frac{t_1 t_2^T}{(\text{nnz}(t_1 + t_2))} & ; (\text{nnz}(t_1 + t_2)) > 0 \\ 0 & ; \text{otherwise} \end{cases}$
Subset	$\text{dist} = \begin{cases} 1 & ; t_1 == t_2; \\ 0 & ; \text{otherwise} \end{cases}$
Hamming	$\text{dist} = \sum_{i=1}^{|t_1|} |t_{1,i} - t_{2,i}| / |t_1|$
Cosine (if embedded)	$\text{dist} = t_1 t_2^T / (\|t_1\|_2 \|t_2\|_2)$
Hyperbolic (if embedded)	$\text{dist} = \text{arcCosh}(-t_1 t_2^T)$

The t_1 and t_2 correspond to two rows, the nnz represents the count of non-zero elements in a given row vector. Note that the considered vectors are binary in all but the cosine and hyperbolic cases (last two rows)

this information is used for updating feature weights as follows. Let K represent the set of considered nearest neighbors. Let T-DIFF represent the mean of the target distances TAR-DIFF. Let D-DIFF represent the mean of the (descriptive) distances to neighbors (DES-DIFF), as also considered for the MLC case, i.e. the absolute difference between the selected instance r_i and the mean of the nearest neighbors. More formally

$$\text{T-DIFF} = \mathbb{E}[\text{dist}(t_i, T[n \in K])] \quad \text{and} \quad \text{D-DIFF} = \mathbb{E}[\text{dist}(r_i, E[n \in K])],$$

where $X[n \in K]$ keeps only the rows of the matrix X that correspond to the neighbors $n \in K$.

We further define

$$\text{TD-DIFF} = \mathbb{E}[\text{TAR-DIFF} \odot \text{DES-DIFF}],$$

and the weight update can be defined as:

$$w[j] += \frac{\text{TD-DIFF}}{\text{T-DIFF}} - \frac{\text{D-DIFF} - \text{T-DIFF}}{1 - \text{TD-DIFF}}. \tag{1}$$

The weight update concludes the ranking for multi-label classification.

3.8 Theoretical analysis

We next discuss the relevant theoretical aspects of ReliefE, ranging from computational complexity analysis (both time and space) to the implications of the adaptations considered.

3.8.1 Time complexity

The time complexity of ReliefE can be studied with respect to the two main modes of function—multi-class and multi-label classification.

Dimensionality estimation. The dimensionality estimation step, as optionally considered in this work, requires pairwise distance computation between the instances. Thus, $\Theta(|v|^2 \cdot |F|)$ operations are required, where v represents the indices of the samples for dimension estimation, and $|v| < |I|$, as discussed in Sect. 3.5. Note that if all instances are considered, the complexity rises to $\Theta(|I|^2 \cdot |F|)$.

Manifold projections. Learning low-dimensional representations represents one of the computationally more intensive parts of ReliefE. Following (McInnes et al. 2018b), the UMAP's complexity can be split into two main parts. First, approximate nearest neighbor computation was shown to have empirical complexity of $\mathcal{O}(|I|^{1.12} \cdot |F|)$ (Dong et al. 2011). However, in the sampling limit, if all instances are considered, the computational complexity is equal to that of pairwise comparisons—$\mathcal{O}(|I|^2 \cdot |F|)$ The optimization of embeddings requires additional $\mathcal{O}(k \cdot |I|)$ steps, where k is the number of nearest neighbors (a hyperparameter). Overall complexity is in the worst case thus $\mathcal{O}(|I|^2 \cdot |F|)$. Note that the proposed cyclic sampling scheme (Sect. 3.5) implies $I \rightarrow v$ for all cases in this paragraph.

Multi-class. Given a fixed number of samples s, ReliefE traverses each of the classes $|T|$, and for each one performs the sampling. The adaptive neighbor selection scheme does not cost any additional time w.r.t. $|I|$, as the distances are already computed. The feature update step requires $\mathcal{O}(|F|)$ operations, for each neighbor. The complexity of the original, re-implemented ReliefF is thus $\mathcal{O}(|I| \cdot |F| \cdot s)$ (Robnik-Šikonja and Kononenko 2003). The absMean update does not change this complexity, however, when adaptive scoring is considered, distances to the class members need to be sorted. We re-use the sorted indices of top neighbors to obtain closest distances, thus no additional time is spent on sorting. If ReliefE is considered, the complexity needs to be adapted for input dimension estimation, as well as lower dimension in which distances are computed. The final complexity is thus: $\mathcal{O}(|v|^2 \cdot |F| + |I| \cdot d \cdot s)$, where d is the dimensionality of the embedding. Assuming the "empirical complexity" from the previous paragraph holds, the multi-class complexity can also be stated as $\mathcal{O}(|v|^{1.12} \cdot |F| + |I| \cdot d \cdot s)$

Multi-label. The complexity of multi-label classification needs to additionally account for the distances computed between the target instances. Effectively, $\mathcal{O}(|v|^2|F| + s \cdot (|I| \cdot d_F + k \cdot d_T))$ operations are required, where d_T and d_F correspond to embedding dimensions of the input and output space – for each sample, first distances need to be computed between the instances to that sample within the input space ($|I|$). Once top k nearest instances are identified, the distances of the target

	instance to these k other target instances are computed in d_T space.
Down-stream ranking.	Commonly, Relief algorithms operate in the raw feature space, however, as ReliefE operates via embedding-based distance computation, we consider the option that embeddings are *pre-computed*. This is possible due the fact that many contemporary embedding algorithms *refine* the representation, once the new data is obtained, and do not (necessarily) re-compute the embedding for each new instance. In this case, the initial complexity of $\mathcal{O}(|v|^2 \cdot |F| + |I| \cdot d \cdot s)$ reduces to $\mathcal{O}(|I| \cdot d \cdot s)$.

3.8.2 Space complexity

The proposed implementation of ReliefE in comparison with, e.g., state-of-the-art Python-based implementations (as found in Urbanowicz et al. 2018) operates easily in very high-dimensional, sparse vector spaces. In practice, we adopt the CSR formalism for matrix representation. Here, a sparse matrix is stored as three arrays, a data array, a pointer array and an index array. All three code for non-zero entries in the input space. Note that such representation is not optimal for dense matrices, as it results in some (minor) space overhead. This design decision means that every computationally-intensive operation that is part of ReliefE was re-written with handcrafted CSR-friendly, Numba-compilable methods. More formally, let nnz correspond to the number of non-zero elements in a given matrix. The space complexity of ReliefE can thus be stated as: $\mathcal{O}(\max(\mathrm{nnz}(F), \mathrm{nnz}(E_F)) + \max(\mathrm{nnz}(T), \mathrm{nnz}(E_T)))$.

The complexity thus depends on the relation between the embedded space and the input space, which can be very context-dependent, however very low-dimensional embeddings normally do not result in space overhead, and neither do highly sparse input matrices. More formally, if $\mathrm{nnz}(F) \geq |I| \cdot d$ or $\mathrm{nnz}(T) \geq |I| \cdot d$,

the embedded space will require less (or equal) memory. Note that T, corresponding to a potentially very sparse output space, is similarly considered as a sparse matrix, meaning that classification problems with very high-dimensional target spaces can also be considered, which is to our knowledge one of the first such Python-based, user-friendly implementations. As dimensionality estimation only requires the two closest neighbors, we do not keep all others in memory, the space complexity becomes linear, i.e., $\mathcal{O}(|I|)$ (in fact, exactly $2 \cdot |I|$). We empirically discovered that UMAP's memory requirements are the main space bottleneck, and, based on the evaluation on the larger data sets, require $\mathcal{O}(|I|^2)$ (empirical) space. Such complexity potentially arises from the dense computational graph derived by UMAP. This observation led us to introduce the representative (cyclic) sampling scheme, which reduced this complexity to $\mathcal{O}(|v|^2)$, making ReliefE executable even on an off-the-shelf computer (Lenovo Carbon X1). Note that the number of samples is lower-bounded by the number of classes or unique label sets.

3.8.3 absMean update step and its implications

Compared to the original ReliefF, one of the proposed modifications implemented in ReliefE is the comparison of a given instance directly to the average nearest neighbor. We believe that this approach is advantageous in two ways.
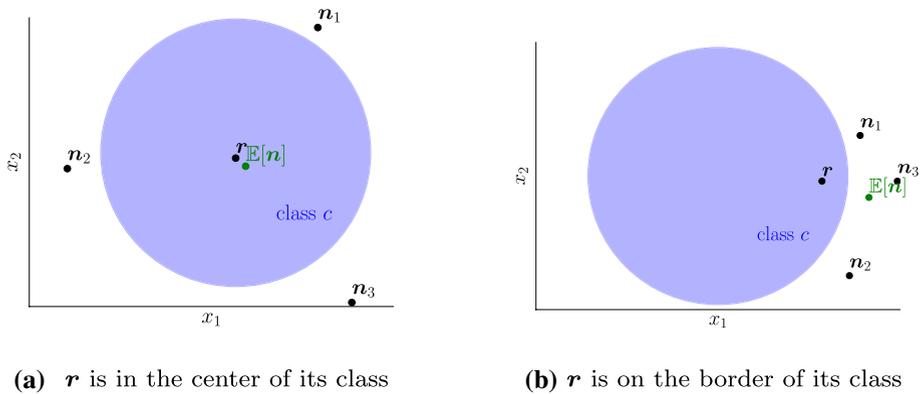
(a) r is in the center of its class   **(b)** r is on the border of its class

Fig. 3 Updating weights with the absMean approach. The $n_{1,2,3}$ represent the instance **r**'s neighbors

First, as shown in Fig. 3a, if a sample r that is far away from the class border is chosen, we cannot capture the local structure of the data in the other classes, so such samples r should not influence the updates considerably. This is not the case in the standard ReliefF, since the differences in feature values are necessarily large. This is overcome by computing the average neighbor first, and then updating the weights.

Second, when the sample r is close to the border (Fig. 3b), averaging neighbors results in correctly detecting that the general direction of the neighbors should be perpendicular to the class borders when the number of samples goes to infinity. For example, in the situation depicted in Fig. 3b, only n_1 should be rewarded. Again, computing the mean neighbor $\mathbb{E}(n)$ first, brings us closer to the optimal direction. The reduction of noise can be also proven by using the triangle inequality, $\frac{1}{k} \sum_{j=1}^{k} |n_i^0 - n_i^j| \geq |n_i^0 - \frac{1}{k} \sum_{j=1}^{k} n_i^j|$, from which it directly follows that this approach results in smaller weight updates.

3.8.4 Adaptive neighbor selection and its behavior

The considered adaptive neighbor selection attempts to reduce the number of hyperparameters by one (k), potentially saving $\mathcal{O}(k)$ optimization iterations, should this parameter be tuned. Furthermore, by considering neighbors, potentially relevant for a given instance, less noise is considered during the weight update step. For example, assume $k = 7$, with only three other instances very close and the remaining four much further (by a large margin). The latter 4 instances will impact the weight update significantly, as the average distance will be heavily biased towards their mean, and thus potentially not representative of the close neighborhood of a given instance that naturally appears in the data. A visualization in such a situation is shown in Fig. 4.

In both panels ((a) and (b)), the outer circle represents the neighbourhood for a hard-coded value of k. In Fig. 4a, very distant instances are also considered for the update (e.g., from n_3 onward) and the adaptive estimation only selects the closest neighbors (green). However, in Fig. 4b, all instances are very close, thus the value of k is equal to the automatically selected choice.

This follows the intuition behind the Relief family of algorithms, where an instance is compared to its slight perturbations. Another downside of having k fixed, is that taking into account more distant nearest neighbors would (on average), increase the importance of more

Fig. 4 Adaptive neighbor selection. The $n_{1,2,3,...}$ represent neighbors

(a) Dispersed neighborhood **(b)** Local neighborhood
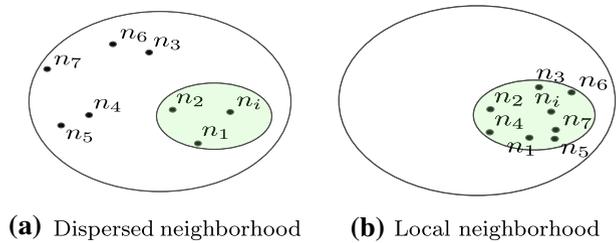
noisy features, since the distance values directly influence the importance. Irregularities in distance distribution were shown to hold for many real-life data sets, see for example the assumptions and their implementation in Dong et al. (2011). Finally, as ReliefE operates in a latent, low-dimensional space, obtained by instance similarity comparison, comparison to the closest instances only is potentially meaningful.

3.8.5 Parallelism aspects

The proposed implementation exploits the Numba framework for just-in-time compilation (Lam et al. 2015). Numba offers parallelism at the level of individual methods that get compiled, meaning that the proposed implementation offers parallelism at the level of weight updates. During compilation, parts of the code that are sensible to compile get detected *automatically*. Many operations such as scalar-vector addition and similar can easily be parallelized. With auto-parallelization, Numba attempts to identify such operations in the ReliefE weight update step, and fuses adjacent ones together, to form one or more kernels that are automatically run in parallel. In practice, we observed that such auto-parallelism does not necessarily offer superior performance in terms of speed. However, it represents an elegant, array-level parallelism detection which, when improved/updated, shall speed up the execution time even more. We omit the discussion regarding different spaces considered during ReliefE to "Appendix 1".

3.8.6 How powerful is ReliefE?

Throughout this paper, we propose and demonstrate the utility of ReliefE when tabular data is considered. However, as ReliefE requires merely the *representations* of instances (training or target), the proposed approach generalizes well beyond tabular data with a single adaptation: the embedding method needs to be suitable for the considered data type. For example, if an instance is described by an ordered list of graphs, the plethora of graph embedding methods (Goyal and Ferrara 2018; Mežnar et al. 2020) could be used to prioritize the graphs based on their (learned) representations. Similarly, ReliefE could be adapted for learning in the context of relational data bases, via Wordification (Perovšek et al. 2015) and other propositionalization-like algorithms.

Table 2 The properties of the considered multi-class classification data sets

Data set	Instances	Features	Classes	Proportion of non-zero entries
chess (Shapiro 1984)	3196	38	2	0.726558
biodeg-p2-discrete (Džeroski et al. 1999)	328	61	4	0.107357
optdigits (Alpaydin and Kaynak 1998)	5620	62	10	0.528117
madelon (Guyon et al. 2005)	2000	500	2	0.999999
php88ZB4Q (Anguita et al. 2013)	10299	561	6	0.999860
pd_speech_features (Sakar et al. 2013)	756	753	2	0.995294
dlbcl (Armstrong et al. 2002)	77	7070	2	0.997388
tumors C (Pomeroy et al. 2002)	60	7129	2	0.995722
AP_Ovary_Kidney (Stiglic and Kokol 2010)	458	10935	2	1.000000
ohscal.wc (Han and Karypis 2000)	11162	11465	10	0.005270
genes (Weinstein et al. 2013)	801	20531	5	0.857824

The last column denotes the proportion of non-zero elements in the data table

4 Empirical evaluation setting

Our empirical evaluation of ReliefE consists of many sub-studies, and can be summarized as follows. First, we discuss the evaluation of ReliefE against state-of-the-art ranking algorithms on eight multi-class classification data sets. Next, we present the empirical evaluation setup where ReliefE's capabilities are shown on nine multi-label classification data sets. Finally, we conducted a series of experiments where we investigated in more detail the convergence and time performance. We conclude this section by describing the Bayesian and frequentist approaches, to aid understanding of the results.

4.1 Multi-class classification data sets

Multi-class classification remains one of the most widely adopted forms of learning. Here, the input space is associated with a single, integer-valued vector, where each instance can belong to one of the many possible classes. In this work, we consider a wide spectrum of data sets, summarized in Table 2.

The data sets are from multiple domains, incl. biological, social and other domains (e.g., chess). The data sets are of different dimensions, in terms of the numbers of rows and also columns.

4.2 Multi-label classification data sets

Feature ranking for multi-label classification remains an active research area. Many of the approaches considered in the previous section (multi-class classification) are not able to handle the multi-label setting, where a single instance can belong to many classes simultaneously. Such a setting, for example, naturally emerges when gene

Table 3 The properties of the considered MLC data sets

Data set	Instances	Features	Classes	Proportion of non-zero entries
delicious (Tsoumakas et al. 2008)	16105	1000	983	0.500000
imdb (imdb dataset 2010)	120919	1001	28	0.019363
medical (Pestian et al. 2007)	978	2898	45	0.500000
bibtex (Katakis et al. 2008)	7395	3672	159	0.500000
Education1 (Ueda and Saito 2003)	12030	27534	33	0.004059
Health1 (Ueda and Saito 2003)	9205	30605	32	0.003555
Entertainment1 (Ueda and Saito 2003)	12730	32001	21	0.004552
Science1 (Ueda and Saito 2003)	6428	37187	40	0.004659
Social1 (Ueda and Saito 2003)	12111	52350	39	0.002949

The last column denotes the proportion of non-zero elements in the data table

function prediction is considered—a single gene is associated with many functions and contexts. The considered multi-label data sets are summarized in Table 3.

Similarly to the multi-class setting, we selected data sets from various domains to maintain diversity. Note that multiple repetitions of 10 fold cross validation were needed to perform Bayesian comparisons.

4.3 Additional experiments and statistical evaluation of results

For MCC, logistic regression with its default parameters was used as the learner. The first reason for this choice is the fact that this very learner is commonly used to evaluate the quality of a given data representation (in our case a subset of the feature space), and is known to be sensitive to noisy features. The second reason is computational: With all repetitions required for Bayesian analysis, additional grid search would be out of reach as it could further increase the computational time beyond reasonable capabilities. For the MLC setting, the default random forest parametrization was used, as it has been previously shown to perform competitively/well in such setting. Throughout the experiments, we set the regularization term of logistic regression (C) to one, the default value (Pedregosa et al. 2011). For multi-label classification, we considered the RandomForest classifier with default settings as set in Pedregosa et al. (2011).

As we consider either multi-class or multi-label problems, we compute either *relative* F1 or micro-averaged relative F1 scores, defined as:

$$\mathrm{rF1}(f) = \frac{\mathrm{F1}_f}{\mathrm{F1}_{f=|F|}},$$

where F1 is the harmonic mean of precision and recall, and f is the number of features. The macro rF1 is defined in the same fashion. Considering relative performance offers direct insights into how performant a given ranking is with how many top-ranked features. Note that by considering relative performance, it can be directly observed when the feature ranking algorithm identifies a ranking that outperforms the situation where all features are

considered—a reasonable baseline. We performed ten fold stratified cross-validation ten times, as required for the statistical analysis discussed next.

In order to summarize the overall performance of a given ranking, we believe taking into account the ranking's quality over all possible values of top f features needs to be considered. Hence, we introduce the area under rF1 (AUrF1), i.e., the integral of rF1 normalized by the number of considered top f rankings (to be more comparative across data sets), where we numerically integrate with the Simpson's method.

Recent criticisms of the frequentist non-parametric comparison of multiple classifiers (Demšar 2006) has given rise to a novel spectrum of Bayesian t-tests, that directly offer insight into a probability space corresponding to the differences in algorithm performance (Benavoli et al. 2017). In this work we adopt the hierarchical t-test, which is capable of comparing pairs of classifiers. The hierarchical Bayesian t-test is used to assess the probability of observing a given difference in performance between a pair of classifiers. As noted by Benavoli et al. (2017), it requires that e.g., ten repetitions of ten fold cross validation need to be considered in order to reliably fit a hierarchical model. The approach attempts to model the probability of observing a given difference in performance between a pair of classifiers, which can be in favor of either of the classifiers or undetermined—practically equivalent (rope region). The plotted results are given in the form of triangular schemes, where each point represents a sample from the posterior distribution. Such samples, when aggregated, directly represent a probability of observing a given state (in this case difference between the classifiers). We set the rope region to 5%—if the difference in quality between two rankings is less than 5%, they are considered equal. The remaining setting is the same as in the original paper (Benavoli et al. 2017), we compare the top ranking for each fold. For a given pair of ranking algorithms, the pairwise Bayesian tests were performed on the data sets common to both algorithms. Finally, results of time performance are presented in computation time (in seconds) diagrams with standard deviations. Such a comparison is not necessarily informative/useful when multiple classifiers are simultaneously considered, thus we also offer the results in the form of average rank diagrams (Demšar 2006). We believe that having both local and global insights into the relations between classifiers their differences are easier to study, even though looking at the classifier ranks alone can be misleading (Benavoli et al. 2017).

4.4 Considered implementations and baselines

We next discuss the implementations considered. For multi-class classification, the considered Relief variants were MultiSURF, MultiSURFstar, ReliefF, all from the scikit-rebate library (Urbanowicz et al. 2018). We also used RandomForest (RF)-based importances (Genie3) and Mutual information (MI)-based ones (Pedregosa et al. 2011). The multi-class Relief variants that are the original contribution of this work include: ReliefE, ReliefE-absMean, ReliefE-adaptive and ReliefE-absMean-adaptive. The suffix *adaptive* denotes the use of an adaptive threshold and *absMean* the use of absMean update step.

Multi-label classification is not supported (at all) in scikit-rebate (Urbanowicz et al. 2018), and thus we considered the multi-label variants of ReliefE and ReliefF (re-implemented in this work with Numba) with all of the possible distances given in Table 1. We emphasize that when multi-label distances are considered, only the cosine and hyperbolic

Table 4 Computational complexity of feature importance estimation

Algorithm	Time complexity	Space complexity
ReliefE	$\mathcal{O}(|v|^2 \cdot |F| + |I| \cdot d \cdot s)$	$\mathcal{O}(|v|^2)$
ReliefF	$\mathcal{O}(|F| \cdot |I|^2)$	$\mathcal{O}(|F|)$
Random Forest	$\mathcal{O}(t \cdot |F| \cdot |I| \cdot \log^2 |I|)$	$\mathcal{O}(|I| + |F|)$
Mutual Information	$\mathcal{O}(|F| \cdot |I|)$	$\mathcal{O}(|I|)$

For Relief algorithms, we used $s = |I|$. The t corresponds to the number of trees

distances operate on target space embeddings (the other distances do not). The computation of these distances is also more efficient.

Note that all versions of ReliefF, implemented or re-implemented in this work,[3] natively operate on sparse spaces, which is on its own a contribution of this work. In terms of sparsification, we set the sparsification threshold to 0.15, meaning that if a matrix's density is higher than 15%, it is sparsified with the proposed procedure (there are many of such matrices amongst the considered data sets). Detailed results of investigating the ablation of the considered data sets' (induced) sparsities are given in "Appendix 2". Similarly, the behavior of the adaptive k statistic was also studied in more detail in "Appendix 3". Further, v (the sample for intrinsic dimension estimation) was set to 2048. The dimension number was set so that the algorithm runs normally on an off-the-shelf-computer (Lenovo Carbon X1) even for larger data sets. Thus, if a given data set consisted of more than 2048 instances, a representative subset of 2048 instances was considered for estimating the intrinsic dimension and consequent embedding. The UMAP's setting is left to its defaults, with the dimension being set to the estimated one.[4] The value of k is set to 15 for our implementation for ReliefF, and left at its defaults for the baselines. The time and space complexity of the baselines and ReliefE are summarized in Table 4. Note that, even though ReliefF (and its other variants') space complexity is linear w.r.t |F|, their implementations, should they not consider the sparse input structure, in fact require $\mathcal{O}(|I| \cdot |F|)$ space (as found, e.g., in Urbanowicz et al. 2018).

Finally, the considered experiments for multi-label classification consider both Euclidean embeddings, as well as non-Euclidean ones (Poincaré ball).

5 Results

This section presents the results of the empirical evaluation. We begin by discussing the performance comparisons for the task of multi-class classification. We follow on by discussing the results of the experiments on multi-label classification tasks. Finally, we present additional investigations of ReliefE's behavior.

---

[3] Implementation's official repository is https://github.com/SkBlaz/reliefe.

[4] Extensive evaluation of UMAP's capabilities w.r.t. the proposed implementations is beyond the scope of this paper, and is left for further work.
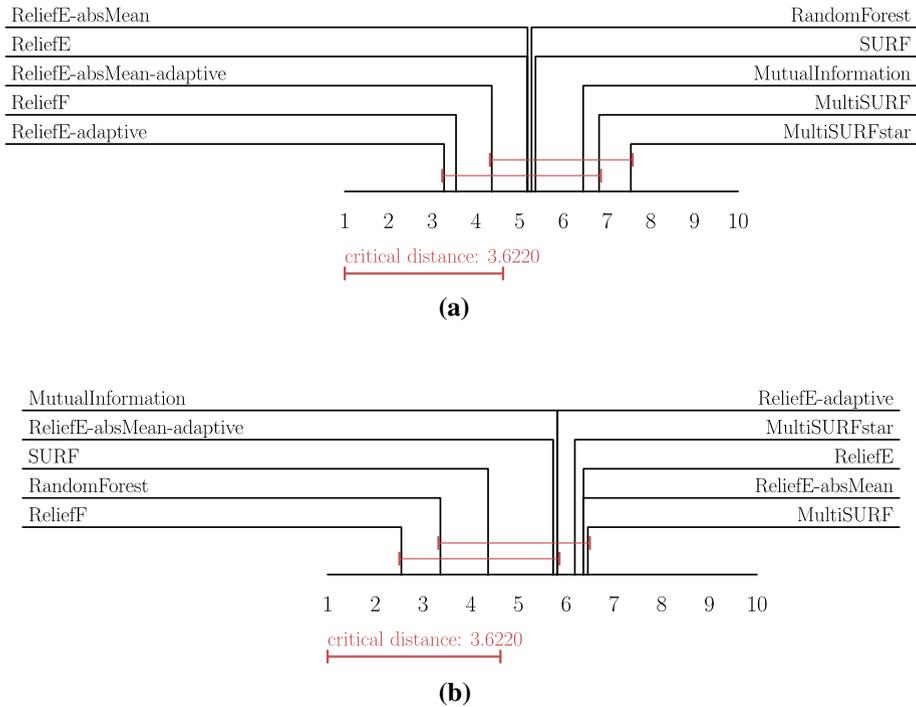
Fig. 5 Max (**a**) and mean (**b**) F1 scores across all feature rankings

5.1 Multi-class classification

We first present two average rank diagrams depicting the relative performance on the different ranking methods for MCC in terms of the quality of the produced rankings, as measured by the corresponding average and maximum F1 scores (Fig. 5a, b , respectively). The diagrams include critical distances, representing the minimum differences in performance that are statistically significant. It can be observed that the ReliefE variants yield the best performing rankings (with lowest average ranks, Fig. 5a), but there are not many such rankings (Fig. 5b). The AUrF1 values ("Appendix 4") indicate that the performances of the top 5 feature ranking algorithms are highly similar (within the confidence interval).

We next present the mean time consumption averaged across data sets. Consistently slower SURF variants of ReliefF can be observed in the rightmost part of Fig. 6a. The average rank diagram is shown in Fig. 6b.

Additional analysis of the proportions of time spent at different parts of the algorithm is presented in "Appendix 5", showing that most time is spent on feature weight updates. Average rank diagrams comparing the rankings in terms of the top 50 and 100 features are given in "Appendix 6".
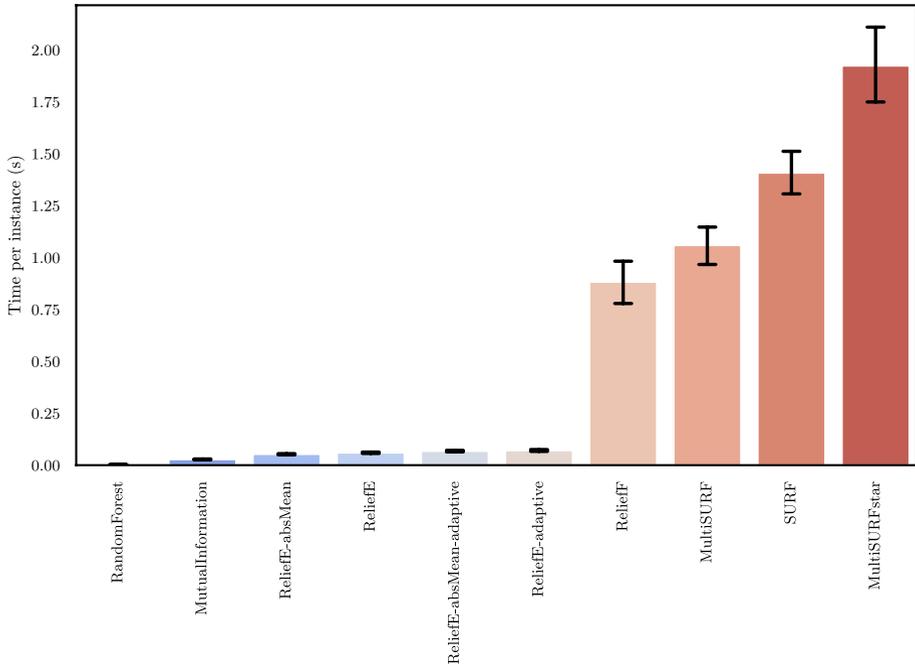
(a) Average absolute running times with standard deviations (in seconds).
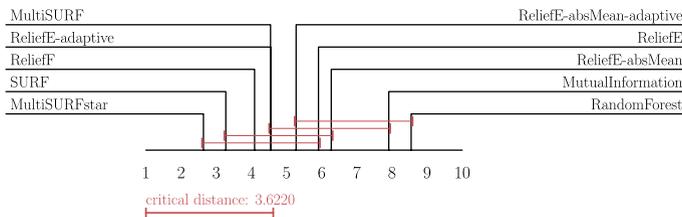
(b) Average rank diagram (times).

Fig. 6 Speed comparison of ranking approaches for MCC. Absolute running times (**a**) show that ReliefE variants perform an order of magnitude (or more) faster. Relative running times are given in terms of average ranks (**b**), where lower ranks mean worse performance, i.e., longer running times

5.2 Bayesian ranking comparison of ranking approaches for MCC

In this section, we present selected Bayesian pairwise comparisons of classifiers' performance. Previously determined relationships, such as the dominance of the SURF branch of algorithms over mutual information were confirmed, and further extended by adding comparisons with the proposed ReliefE branch of algorithms. The comparisons are presented in Figs. 7 and 8.

Each diagram has three main regions (parts of the pyramid). The two bottom regions correspond to the samples associated with the dominance of each of the two algorithms
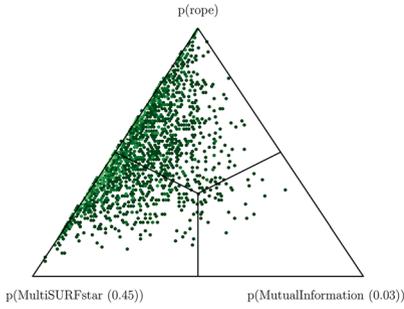
(a) MultiSURFstar vs. MI
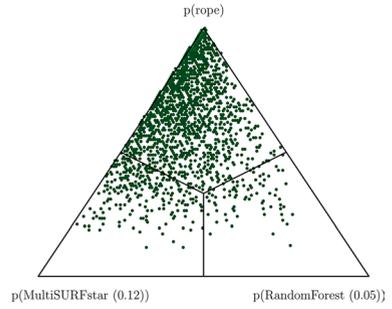
(b) MultiSURFStar vs. RF
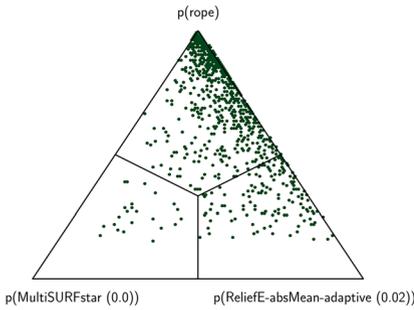
(c) MultiSURFstar vs. ReliefE-absMean-adaptive
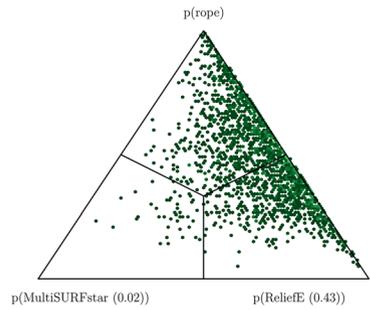
(d) MultiSURFstar vs. ReliefE
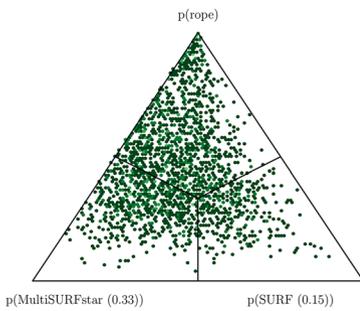
(e) MultiSURFstar vs. SURF
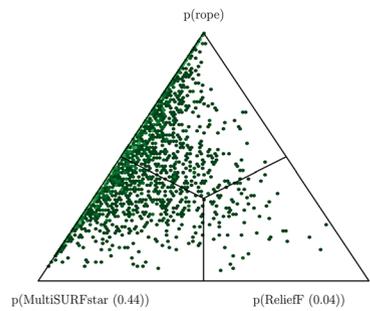
(f) MultiSURFstar vs. ReliefF

Fig. 7 Bayesian comparisons of performance (ranking qualities) between MultiSURFstar and other feature ranking methods for MCC

(a) ReliefE-absMean-adaptive vs. SURF

(b) ReliefE-absMean-adaptive vs. Multi-SURF

(c) ReliefE-absMean-adaptive vs. RandomForest

(d) ReliefE-absMean-adaptive vs. ReliefF
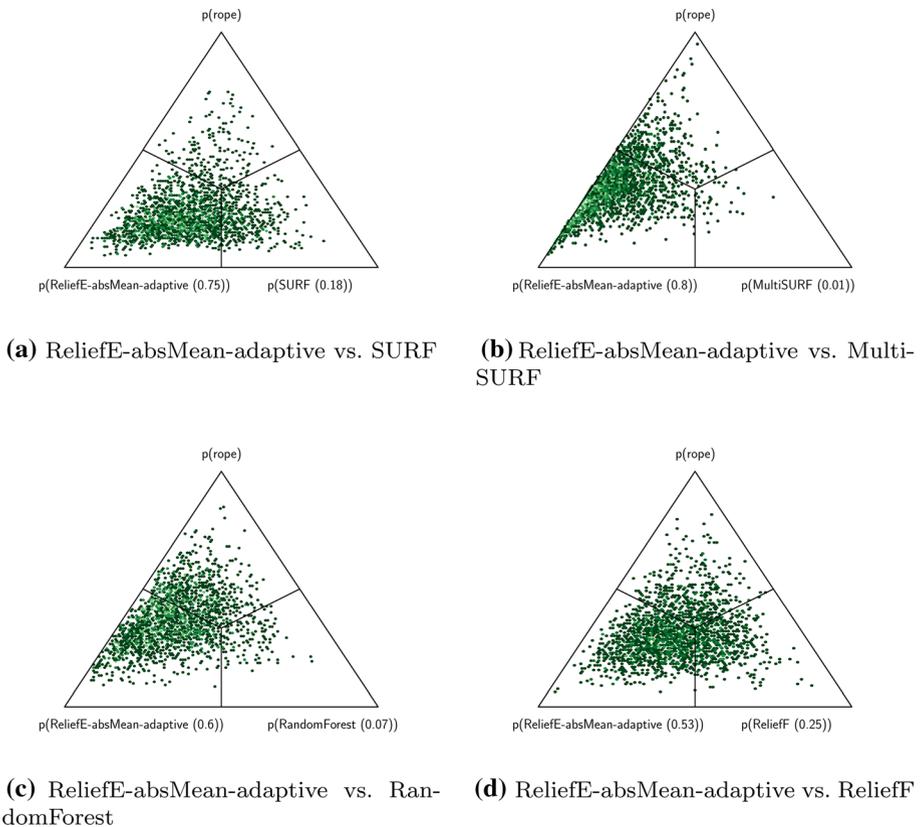
Fig. 8 Bayesian comparisons of performance (ranking qualities) between ReliefE-absMean-adaptive and other feature ranking methods for MCC

compared, and the rope region to the difference space, where the winner is not clearly defined. The probability density directly corresponds to the density of dots in the diagram, thus, the part of the diagram with the highest density implies the most probable situation. Individual (posterior) probabilities are also shown next to each diagram, and denote the probabilities of one algorithm outperforming the other or the algorithms being of similar performance.

The key results of such pairwise comparisons can be summarized as follows. Very few comparisons yield clear winners. In the majority of the cases, when the most competitive methods are considered, less than 50% probability that one of the ranking algorithms dominates is observed, giving no strong evidence for dominant ranking algorithms. This is the case also for the diagrams in Fig. 7.

The visualizations in Fig. 8 show that ReliefE-absMean-adaptive, the implementation proposed in this work, performs on par, or better than many existing, well established approaches such as MultiSURF and RandomForest-based rankings. However, we observe, in the second part of Fig. 8, that ReliefE-absMean-adaptive offers small, albeit incremental win rate when compared against the other methods. With the highest probability (80%), we can claim ReliefE's dominance against MultiSURF, however, the observed probability
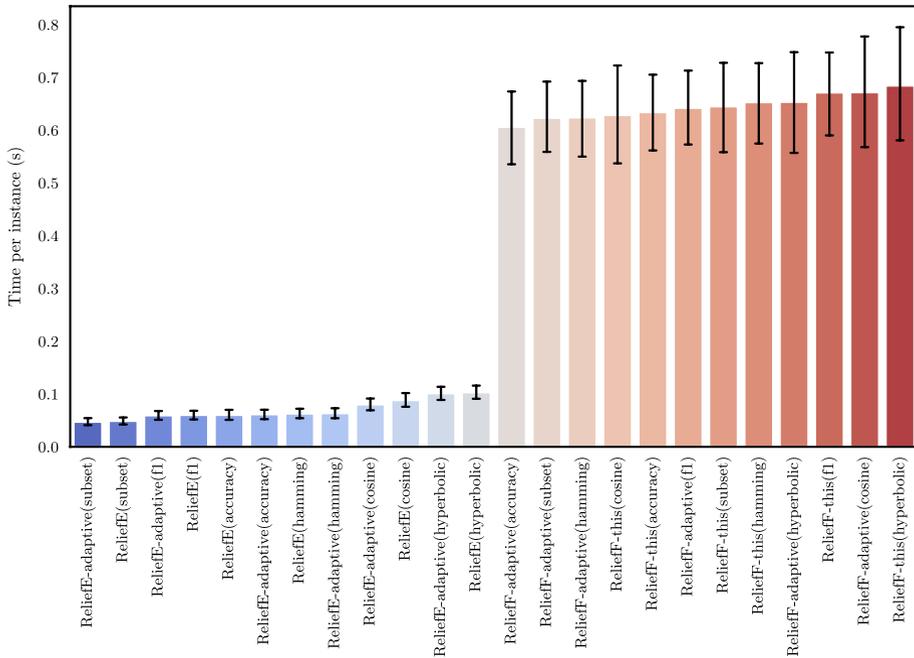
Fig. 9 Running times for the MLC variants of ReliefE (and ReliefF, reimplemented in this work and denoted ReliefF-this)

ratio does not suffice for a significant difference with $> 95\%$ probability (the commonly considered convention). To further study the algorithm performance, we visualize the top f features—rF1 curves and discuss the selected examples—such figures showing in detail the ranking performance of the different algorithms for the selected data sets are given in "Appendix 7". Overall, considering the different statistical approaches to evaluating ReliefE's performance, the results indicate that the method has similar performance to its competitors, but offers up to two orders of magnitude faster ranking computation, which also confirms the theoretical findings from Sect. 3.8.

5.3 Multi-label classification

We next present the results of feature ranking for multi-label classification. For readability purposes, we present the average rank diagrams in "Appendix 8". The time required for the execution of various distance-ranking algorithm combinations is shown in Fig. 9.

The differences in the execution times are apparent. The ReliefE branch (blue) offers more than an order of magnitude faster ranking computation.

The AUrF1 scores, averaged across data sets are shown in Fig. 10.

The best performing ReliefF variants for multi-label classification do not embed the input space. However, the top performant variant employs Euclidean embeddings of the target space, where the distances are computed based on the cosine similarity score. This result indicates multi-label classification can benefit from embedding-based approaches. A
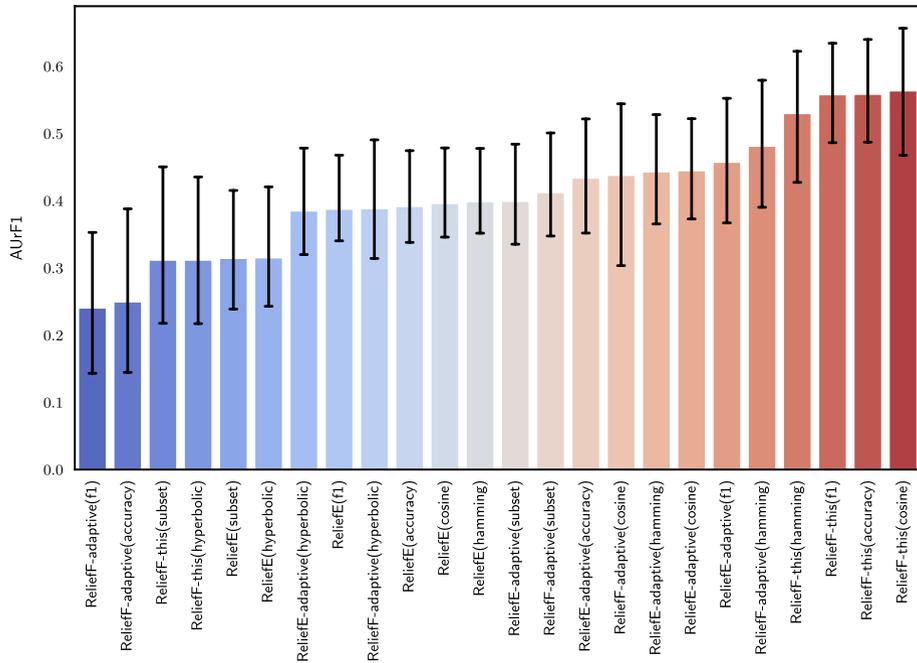
Fig. 10 Area under the relative F1 for different ranking approaches in the context of multi-label classification

case study, where the behavior of various ReliefE variants for MLC is considered in more detail can be found in "Appendix 9".

5.4 Relations between ranking algorithms

We employ the FUJI score, a recently introduced scale-free comparison of ordered positive real-valued lists, to study how different feature ranking algorithms relate to each other. This study employs the same methodology as discussed in Petković et al. (2020) and Škrlj et al. (2020). The considered FUJI scores can, apart from the ranking, also take into account the differences between the elements that are being compared—this is not possible by using, e.g., the Jaccard score. We compare pairs of curves comprised of (rF1,top *f*) tuples, thus effectively comparing the *shape* of the rankings' performance. The results of these comparisons are shown in Fig. 11 for multi-class classification and in Fig. 12 for multi-label classification. The most apparent pattern that emerges when these comparisons is that embedding-based rankings (ReliefE variants) tend to give very similar rankings. This holds for both multi-class and multi-label classification rankings.

5.5 Convergence to the final ranking

Note that in all the examples up to this point, the number of iterations via which the weights corresponding to feature importances were updated was equal to the number of instances (hence the quadratic complexity). Having shown that this setting already offers
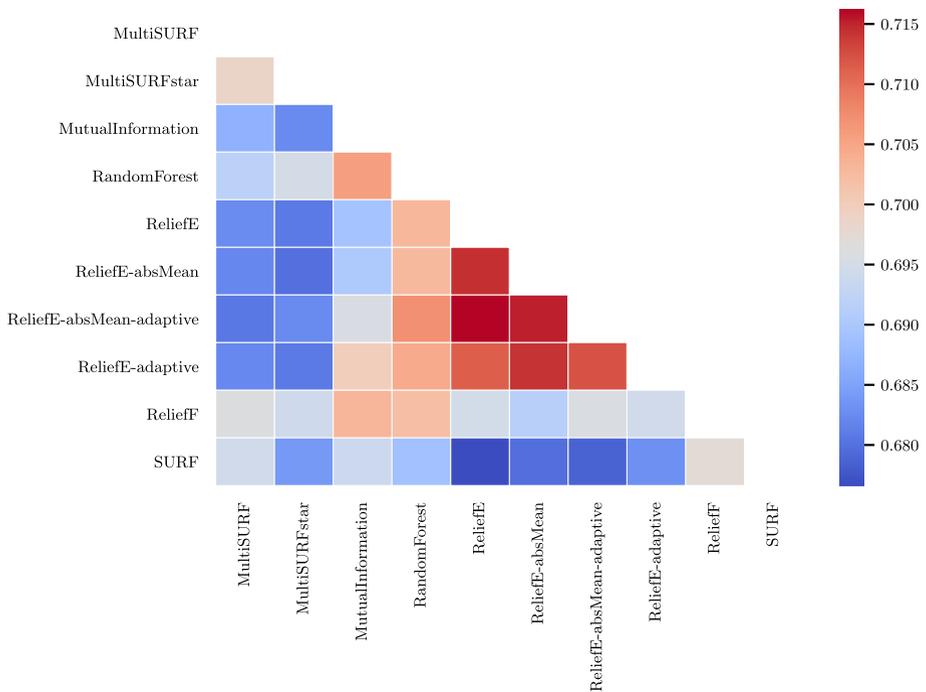
Fig. 11 AUFUJI scores for multi-class rankings. Higher numbers (red colors) mean higher similarity between rankings (Color figure online)

state-of-the-art performance, we further explored how redundant is the iteration process, i.e., what is the minimum number of iterations needed to obtain a similar ranking. We investigated this question on MCC datasets following the approach described below.

For each number of considered iterations, we conducted 100 logistic regression runs building models with up to 100 top-ranked features. We computed the AUrF1 and inspected the curve induced by the obtained series of (top f, AUrF1) tuples. We conducted these experiments for the DLBCL, Tumors C, Biodeg-discrete and chess data sets, with the results shown in Fig. 13. We compared ReliefE-absMean-adaptive with ReliefF as implemented in this work, evaluating each iteration with three-fold cross validation (same splits).

It can be observed in Fig. 13a that the convergence is slower with the ReliefE-absMean-adaptive variant, however, once the performance is achieved, it is no longer impacted by additional iterations. This does not appear to be the case with ReliefF, where a decrease is observed when 32 iterations are considered. Overall, ReliefE-absMean-adaptive offers state-of-the-art performance already after four iterations. A similar situation is observed in the case of Biodeg in Fig. 13c. We also observed that on the Tumors C data set (Fig. 13d), ReliefE-absMean-adaptive was consistently outperformed by ReliefF. Being very high-dimensional, and with only tens of instances, this data set's intrinsic dimension is most likely under-estimated, yielding feature ranking based on representations that loose too much information. The ReliefE branch of algorithms is highly dependent on the underlying embeddings, where construction of high quality embeddings in such data scarce scenarios remains a lively research area on its own.
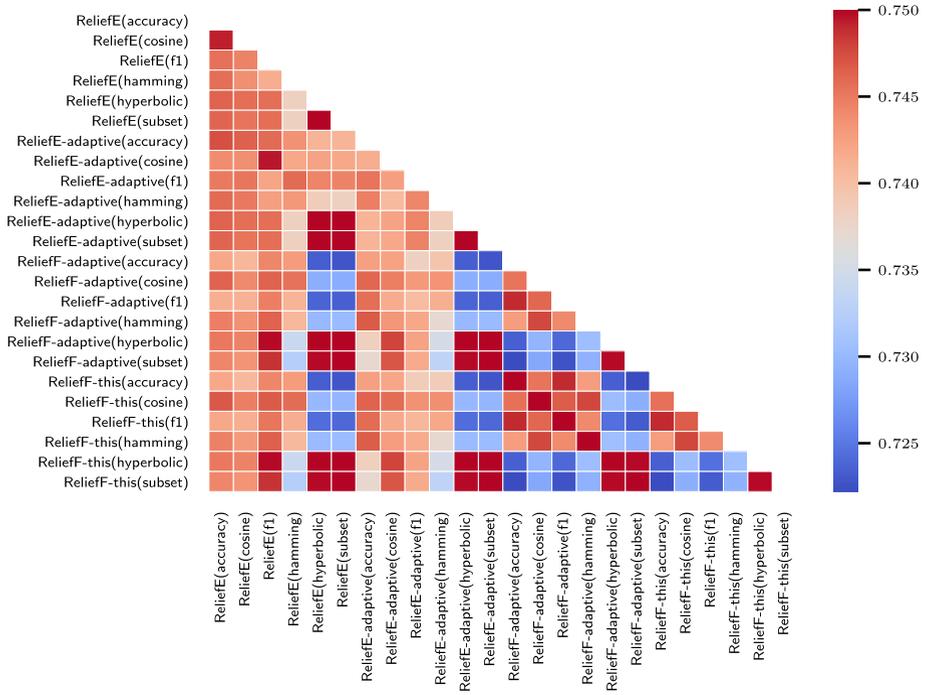
Fig. 12 AUFUJI scores for multi-label rankings. The red block of cells in the upper left part of the triangle corresponds to various variants of ReliefE (Color figure online)

(a) DLBCL

(b) CHESS

(c) Biodeg-p2-discrete

(d) Tumors C
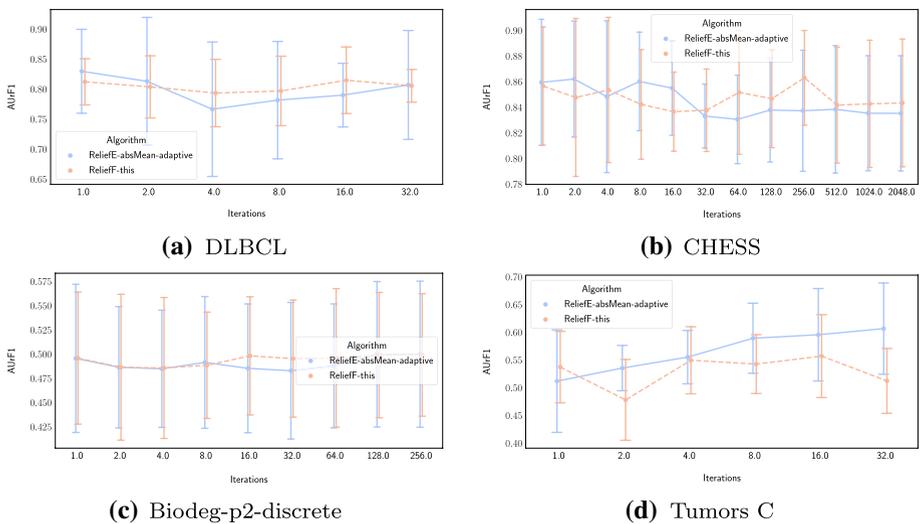
Fig. 13 Impact of the number of ReliefF iterations on ranking quality

Potential speedups by decreasing the number of iterations will be explored in further work. The performance on the chess data set, however, remains consistent for both algorithms—this is a low-dimensional data set, where feature importance estimation via embedded space does not offer notable performance improvements, both with respect to top F1 and computation time.

5.6 Relevant negative results

Even though the paper proposes a promising Relief variant, capable of operating in high-dimensional sparse spaces, many intermediary steps did not perform as expected, and are summarized below:

1. Due to pointer-based storage, using sparse matrix algebra can result in additional overhead which can be significant in large, dense data sets.
2. Running UMAP with spectral decomposition resulted in an unexpected memory overhead. We circumvented this issue with ν, however, the original implementation, once adapted for large scale embedding, could offer an alternative that is more native to UMAP's routines.
3. Employment of Numba's parallel capabilities led to somewhat mixed results. On one hand, trivially parallel routines such as independent looping and similar could easily be adapted to run in parallel, however, when the parallel decorator was employed over the whole ReliefE weight update step, even though all cores were utilized, no notable speedups were observed. Additional study of the intricacies of such decorator-based parallelism is left for further work.
4. When validating our and scikit-rebate's implementations against Weka's ReliefF, it turned out that ReliefF, as implemented in scikit-rebate differs with a somewhat negative effect on performance (as shown in this paper).
5. We did not experiment with detailed typing of the most time-consuming methods, however we believe some of the routines could be, this way, made even faster.
6. The intrinsic dimension algorithm (Algorithm 2) appears to *underestimate* the real dimension, leading to poorer performance in some cases.
7. Embedding target instances in hyperbolic space either works well, or does not work at all. We believe the observed performances are due to the intrinsic geometry of the data, which we will explore in further work.

We next discuss some of the general observations and their implications.

6 Discussion

In this work, we considered extensions of the original ReliefF approach with embedding-based distance computations to both multi-class and multi-label classification settings. We observed that, especially in MLC, embedding the target space can contribute both to lower running time and improve a classifier's performance. The distance that showed the most promising results was based on the cosine similarity, which is widely used when considering embedding-based learning and exploration. The main contribution of this work, the ReliefE ranking approach, is capable of operating via embeddings of both input and output spaces (e.g., in multi-label classification).

In this section, we comment on the obtained results and discuss further implications of ReliefE. We first observe that adaptive neighbor selection empirically performs very similarly to implementations where neighbor selection is hard-coded. This positive results indicate that one hyperparameter less needs to be tuned, should the user not have the computational resources for extensive grid searches. Further, the simple adaptation of the update step to take into account the distance to the mean of the neighbors similarly offered competitive results. One of the possible reasons for such performance is the potential cancellation of noise, as with averaging, especially in the embedding space, a joint representation is obtained that can also carry some information regarding the semantic similarity amongst the neighbors.

Within the proposed ReliefE approach, we also explored how data sparsification can be leveraged to further speed up feature ranking in high-dimensional settings. The sparsification procedure was targeted at larger, higher-dimensional data sets and did not affect smaller data sets as much.

In terms of multi-label classification performance, we observed that the classic adaptation of ReliefF with the proposed adaptive distance and the hamming loss was amongst the best performing options. Interestingly, the variant which used the cosine distance on the target space embeddings was also amongst the top three best performing solutions, indicating that multi-label classification potentially benefits more by considering only the embeddings of the target space instances (and not of instances in the feature space). Similarly, the absMean variant of ReliefF was also amongst the top five algorithms performed, indicating that this aggregation scheme is competitive to the widely accepted averaging followed by the absolute value computation step. The best variant of ReliefE that considered both feature and target space embeddings is ranked poorly, indicating that by embedding the feature space, performance is lost (albeit significant speedups can be obtained): This hints at a trade-off between performance and ranking quality. Of the remaining metrics, the subset and hyperbolic distances were amongst the worst performing ones, indicating that hyperbolic embeddings operate well in rather limited settings, possibly where a hierarchical structure of the target space can be observed.

This work is also one of the first (to our knowledge) to compare the performance curves of different ranking algorithms with the Fuzzy Jaccard Index. We observe that embedding-based algorithms proposed in this work behave very similarly, for both multi-label and multi-class classification. Especially in MLC, two consistent patterns emerge. All ReliefE variants are shown to be very similar to one another (red block in Fig. 12). However, also the hyperbolic and subset versions of ReliefF appear to behave similarly to the embedding-based ones, even though the input space was not embedded in these cases. For multi-label classification (Fig. 11), the ReliefE variants again emerge as the most similar (to one another). However, similarly to the MLC comparison, versions of the adapted ReliefF, as implemented in this work, were shown to yield similar performance curves to ReliefE-based variants.

Following the results of ablation studies, we believe further speedups could be obtained by considering fewer iterations. Current experiments indicate that potentially quadratic speedup could be obtained, as adequate performance was already observed after $\sqrt{|I|}$ iterations in some cases. Further, the number of iterations could also be adapted dynamically, by monitoring the feature ranking scores and detecting convergence before all iterations are carried out.

When studying individual data sets, e.g., DLBCL and opt-digits, we observe that ReliefE offers superior performance at a fraction of the computation time required by

the other methods, indicating that the development of approaches based on ideas introduced in this paper is a sensible research avenue.

In this work, we have evaluated feature rankings based on classification performance obtained by robust learners, such as logistic regression, which have not been fine-tuned. The purpose of such evaluation was to emphasize the effect of feature ranking. However, extensive studies of the interplay between regularization regimes (e.g., L1 vs. L2) and ranking performance could also offer interesting insights into the robustness of rankings, and further, their purpose. For example, a L1 regularized learner could automatically discard large parts of the feature space: Although this would be considered as feature selection (and not ranking), it would potentially offer similar results. We leave this type of experiments for further work.

Similarly, the Bayesian comparisons, involving mostly a state-of-the-art feature ranker MultiSURFstar and the proposed ReliefE algorithm(s), indicate that ReliefE is competitive and many times outperforms MultiSURFstar, even in a probabilistic sense. For example, the probability that ReliefE-absMean-adaptive outperforms MultiSURFstar is more than 30%, with most of the remaining probability density lying in the equal performance (rope) region.

Finally, we discuss several potentially interesting future empirical studies that would represent a non-trivial extension of the proposed work. Detailed analysis of the algorithms' performance with respect to various properties of the data sets could offer additional insights into when to use what type of ranking. We believe that meta-learning could be a promising research venue, as by linking the data sets' properties with suitable algorithms could largely benefit situations where embedding-based ranking is not the best option. Overall, if one optimizes for efficient performance on large, contemporary data sets, ReliefE offers a computationally efficient approach, that could serve as a first step to further study where to invest the remaining computational resources, and whether feature ranking is a sensible approach at all (it might not be for, e.g., image-based data). Similarly, understanding whether the choice of the distance score can be further *transferred* between similar data sets also represents an interesting research direction worth of further study. Overall, the proposed paper provides an empirical, as well as a theoretical foundation for potentially more involved embedding schemes, such as e.g., (variational) autoencoder-based ones. We believe that a relevant factor influencing ReliefE's performance is the quality of the *learned* representation, indicating that another promising research venue could be the investigation of different embedding approaches (this work explores different distances within a single embedding approach, but does not consider different embedding approaches).

7 Conclusions and further work

In this paper, we proposed one of the first embedding-based Relief implementations with both theoretical and practical grounding. We explored whether embedding the input, but also the output space onto a Riemannian manifold prior to feature ranking yields better rankings. The results indicate that, while being significantly faster, embedding-based ranking methods do not consistently outperform the ones that do not use the embeddings. However, we show that they are indeed consistently faster than all other Relief-based ranking approaches.

We also show that for multi-label classification, where additional complexity arises due to multiple label co-presence, ReliefE can offer more stable, and on data sets like Delicious, better performance. Further, we demonstrate that embedding the target label space is beneficial for the final ranking's quality in a multi-label setting. The proposed adaptive neighbor estimation procedure could be further developed in terms of the neighborhood dependence with respect to a given metric. Similarly, the current implementation potentially over-estimates the neighborhood size, which could be due to the nature of the embedded space or the method's bias. Both possibilities are to be explored in future work.

We believe that comparison of feature ranking algorithms should also be considered at the level of their properties and not only their performance. In this work, we show that embedding-based ranking gives rise to a fundamentally different type of rankings, which we believe are worthy of being studied further. To our knowledge, we are the first to perform such large-scale comparisons of a long list of ranking approaches (using, e.g., different similarities in MLC) and take into account also the actual values of importance scores of rankings (through the FUJI score), and not only the feature order.

We also observe that the variants of the original ReliefF, as re-implemented in this work, already offer superior performance to, e.g., the SURF branch of algorithms, indicating that their scikit-rebate implementations have some limitations in terms of numeric stability (and are not adapted at all to handle sparsity).

As further work, we believe the study of non-Euclidean spaces could yield many novel insights, as the target space is frequently of hierarchical nature, implying Euclidean geometry is not sufficiently good for its representation. In this work, we show initial results for embedding on a hyperboloid (Pincaré ball model). However, Lorenzian geometry can also be considered.

Appendix 1: Theoretical considerations of embedding spaces

As many of the recently introduced embedding-based methods tend to replace earlier methods, whilst maintaining the representation power, we believe the comparison of the two mappings, when considered, can be represented as a simple diagram. The example, considered to represent ReliefE's mapping compared to, e.g., that of the standard ReliefF's can be represented as

$$
\begin{array}{ccc}
\boldsymbol{F} & \xrightarrow{\ \phi\ } & \boldsymbol{E} \\
 & {\scriptstyle q}\searrow & \downarrow{\scriptstyle f} \\
 & & \boldsymbol{w}
\end{array}
$$

Here, the initial, real valued feature matrix $\boldsymbol{F}$ is either directly (q), or indirectly (ϕ and f) mapped to the output weight vector $\boldsymbol{w}$. Note that $q : \mathbb{R}^{|I| \times |F|} \to \mathbb{R}^{|F|}$, $\phi : \mathbb{R}^{|I| \times |F|} \to \mathbb{R}^{|I| \times d}$ and $f : \mathbb{R}^{|I| \times d} \to \mathbb{R}^{|F|}$. ReliefE operates under the assumption that the initial ranking can be retrieved via latent space $\boldsymbol{E}$ in two steps (ϕ and f).
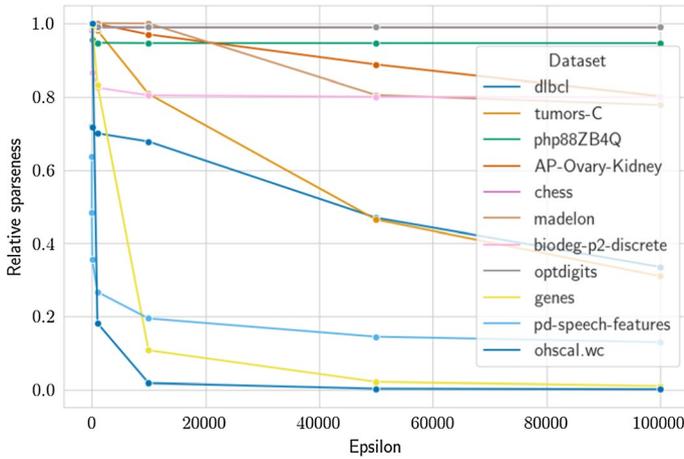
Fig. 14 Dependence of sparsification on approximation error allowed
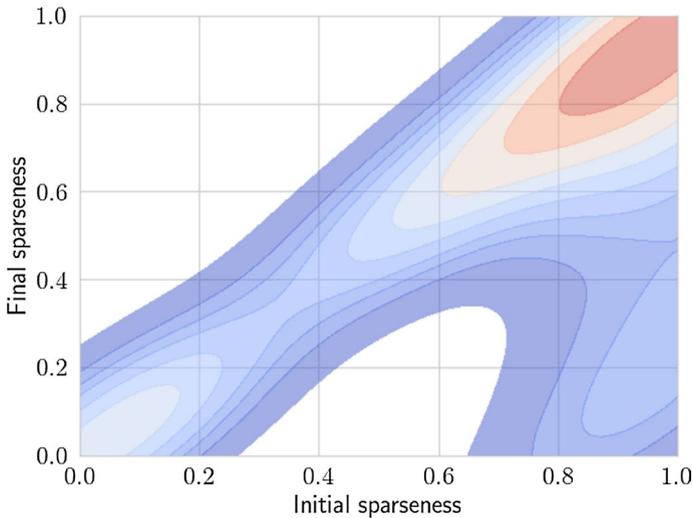

Fig. 15 Kernel density estimation of the relation between the initial and final sparsity

Appendix 2: Ablation study of data sparsity

The considered sparsification procedure is dependent on the parameter *epsilon*, i.e., the approximation error. The study of how different error thresholds impact the final sparsification result is shown in Fig. 14. It can be observed that most of the data sets only get sparsified after a rather large epsilon is permitted. The second ablation explores the relation between the initial data sparsity and the final sparsity, i.e., the sparsity of a given data set after the conducted sparsification procedure. The result is shown as a kernel density plot in Fig. 15.
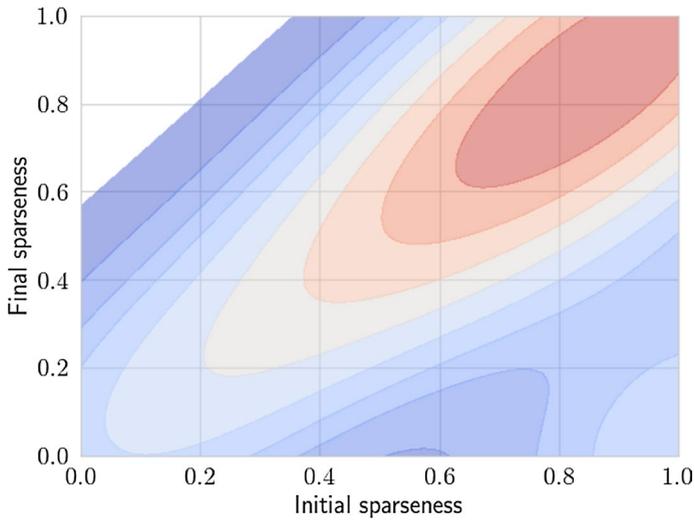
Fig. 16 Kernel density estimation – estimated epsilon values
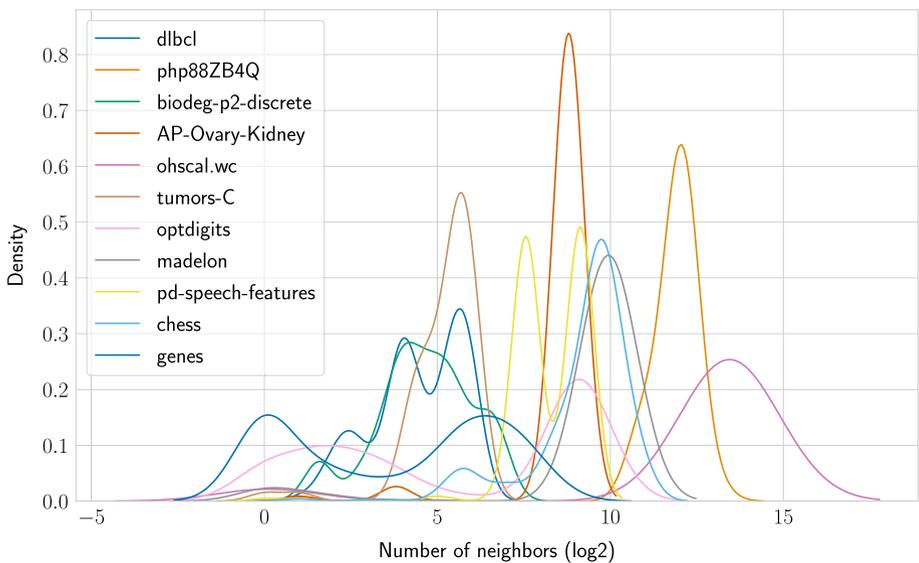
Fig. 17 Density of estimated k values for 100 iterations of ReliefE

The observed result indicates that when the data set is sparse to begin with, the result will be, as expected, similarly sparse. However, the vertical density at the rightmost part of the figure demonstrates that the sparsification procedure indeed yields sparser data, albeit not in all cases. A similar visualization can be produced for the space based on estimated epsilon values, shown in Fig. 16.
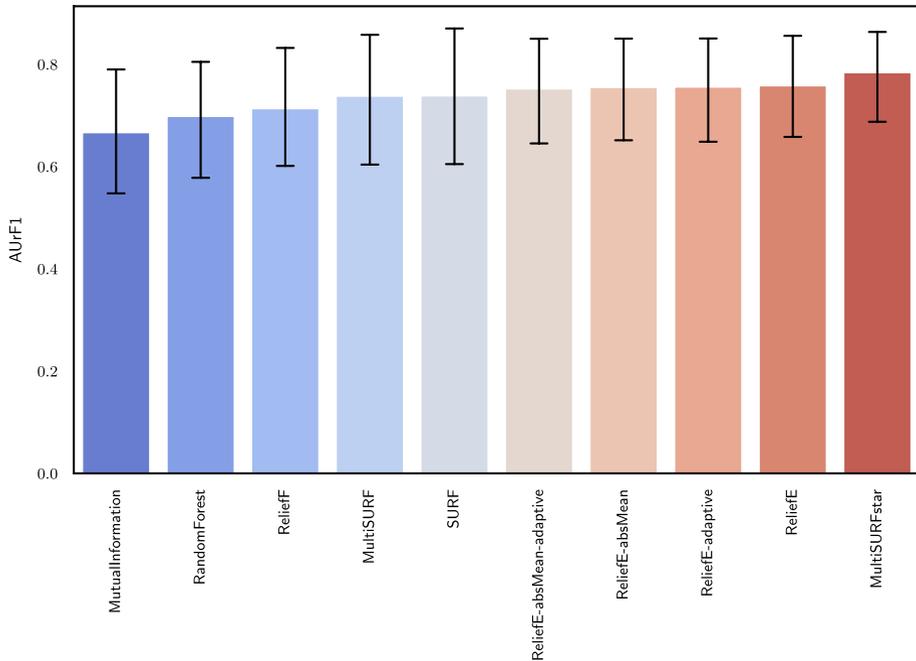
Fig. 18 Area under the relative F1 curve for multi-class classification. All ranking approaches perform similarly with no notable differences. More insights into the relative performance of ranking algorithms are provided by Bayesian tests and FUJI-based comparisons of performance curves

The considered estimate yields a similar landscape sparseness to that obtained via grid-search (Fig. 15), indicating decrease in most cases. However, there are examples where a given data set's density was substantially lowered, such as for example pd-speech-features and biodeg-p2-discrete. The results indicate that the considered estimate could be further relaxed, albeit at the cost of worse approximation of the input matrix, which could negatively impact the final performance.

Appendix 3: Adaptive *k* distributions

In this ablation study, we visualized the distributions of the neighborhoods across all considered MCC data sets. This plot demonstrates that for different data sets, differently sized neighborhoods were identified by the proposed heuristic (Fig. 17).

Appendix 4: Area under the rF1

The AUrF1 scores, averaged across data sets are shown in Fig. 18. It can be observed that the first 5 rankings behave very similarly w.r.t. this measure. Thus, we emphasize other types of comparison, where the differences are more apparent.

Fig. 19 Proportions of time spent on different methods within ReliefE on the p53 dataset. Majority of time is spent on weight update steps (as expected)
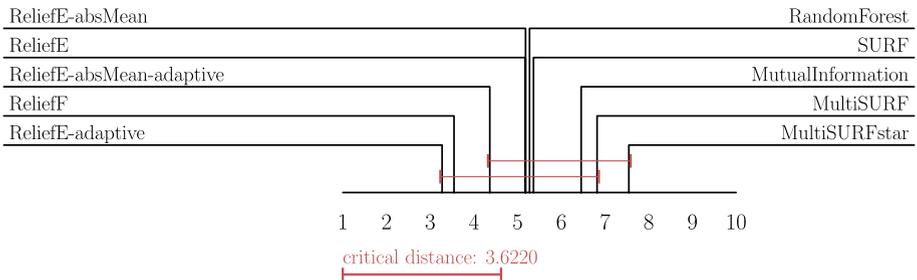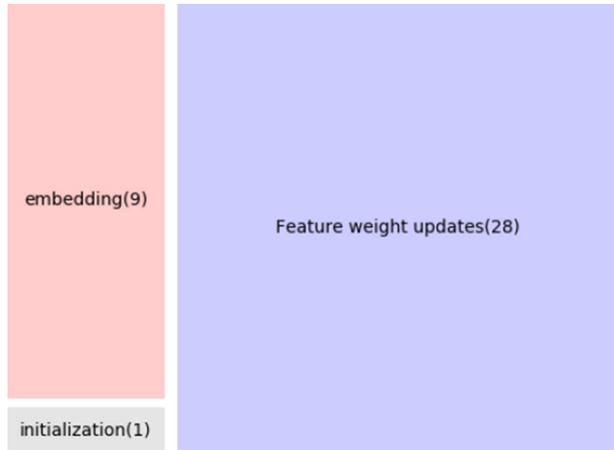
Fig. 20 Max first 100 features. Similarly to the situation with 50 top features, the ReliefE variants, including the adaptive one, perform well for the multi-class classification task
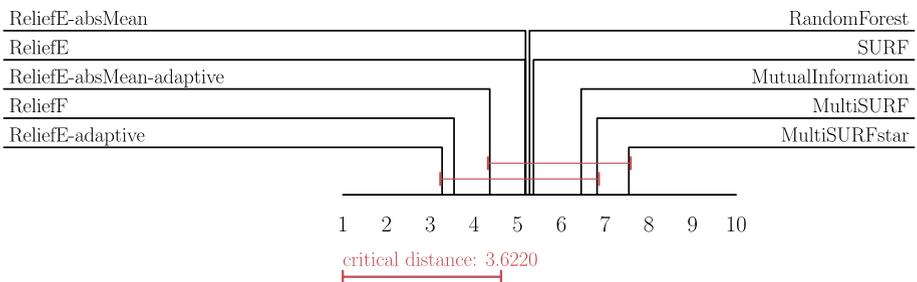
Fig. 21 Max first 50 features. The performance if considering only first 50 features. The adaptive version of ReliefE performs on average the best in this scenario

Appendix 5: Detailed analysis of running time

We additionally studied how different parts of ReliefE impact the total running time. For p53, one of the largest considered data sets, we visualize the proportions in Fig. 19.
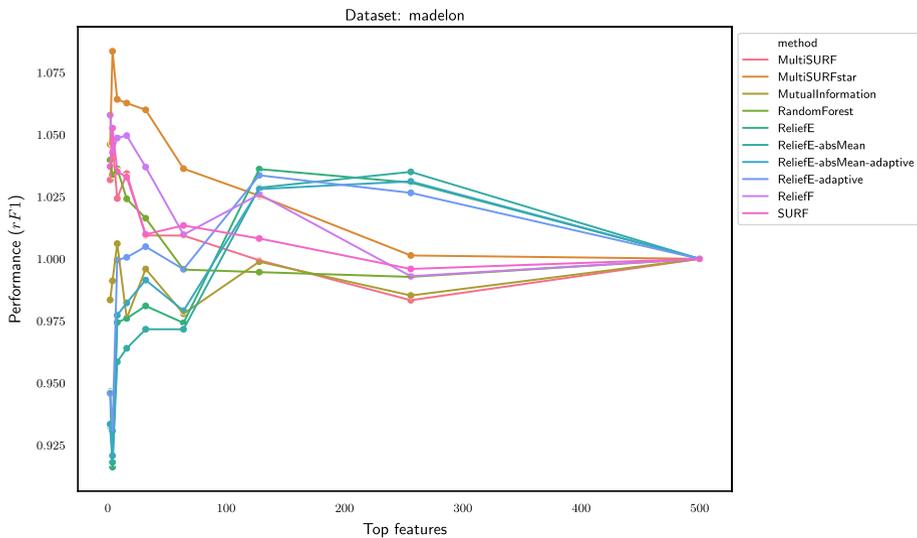
Fig. 22 Madelon performance curves. This result indicates that there exist situations where initially better rankings are obtained via e.g., the SURF branch of the algorithms, however, when considering more features, ReliefE variants are the only ones that find rankings which perform well

Appendix 6: Multi-class classification, additional rank diagrams

This appendix includes additional ablation studies in the form of critical distance diagrams presenting the performance for multi-class ranking (Figs. 20 and 21).

Appendix 7: Multi-class classification—case study with Madelon, DLBCL and genes

This section contains feature rankings, visualized for the Madelon data set, where either the ReliefE or a variant of ReliefF equiped with one of the proposed heuristics shows different behavior (better performance) (Fig. 22.) The visualized performances offer insights into behavior of the algorithms. For example, the ReliefF branch of adaptations (and vanilla ReliefF) peak at less than a hundred features, however, another performance peak where feature ranking is sensible ($rF1 > 1$) is around 250 features, where ReliefE-type algorithms are consistently amongst the best-performing ones.

The power of ReliefE is apparent when considering DLBCL data set (very high dimensional with not many instances). Results are shown in Fig. 23. Finally, the results for the genes data set are shown in Fig. 24. Note how the more time expensive SURF variants were not able to finish in dedicated time. Further, ReliefF is notably worse, requiring more information to detect the relevant signal. On the other hand ReliefE variants perform consistently well.
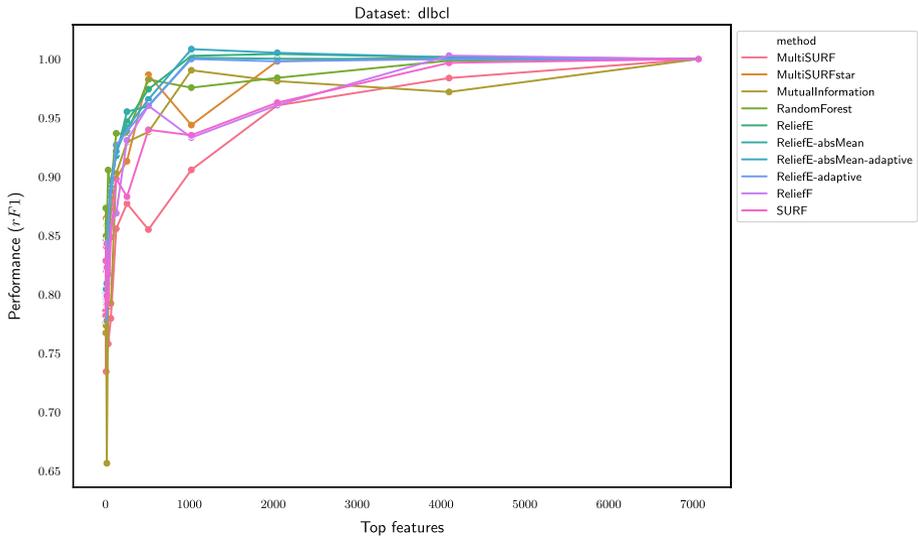
Fig. 23 DLBCL performance curves. The DLBCL is a very high-dimensional data set and reflects the ReliefE's capability to operate with high-dimensional feature spaces
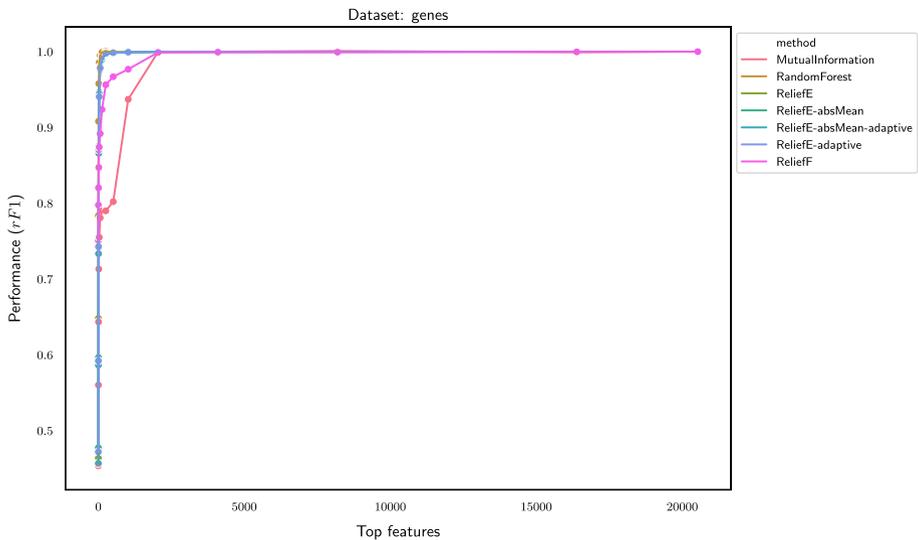

Fig. 24 Genes performance curves. Compared to mutual information (myopic)-based rankings ReliefE performs consistently better (steeper curve at the beginning in the first around hundred features

Appendix 8: Multi-label classification—additional rank diagrams

In this section, we present the average rank diagrams that offer insights into global distribution of the performances when multi-label classification setting is considered (Figs. 25 and 26).
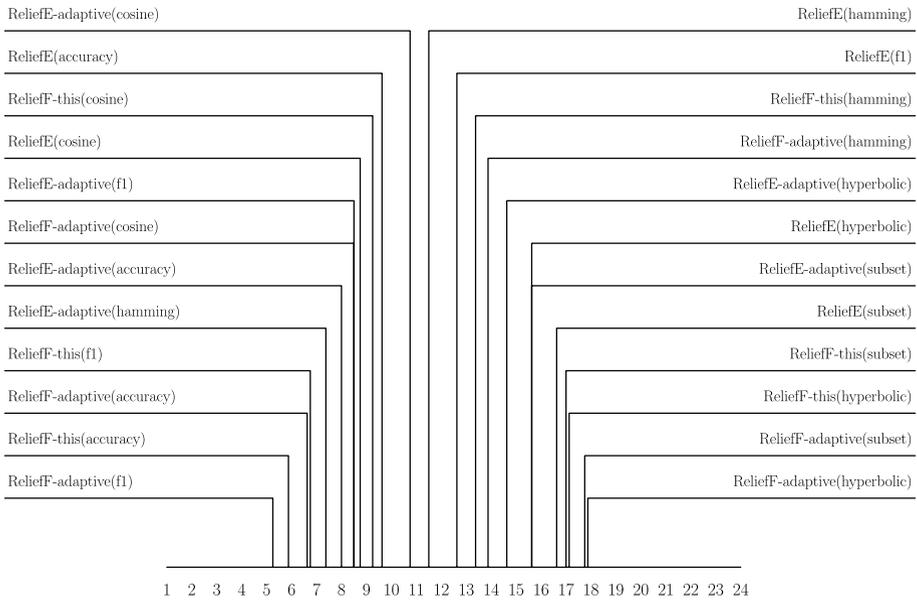
Fig. 25 Max (upper) F1 scores for top 10 highly-ranked features. The results indicate that the adaptive threshold step impacts the placing of the top features (adaptive) version of ReliefF
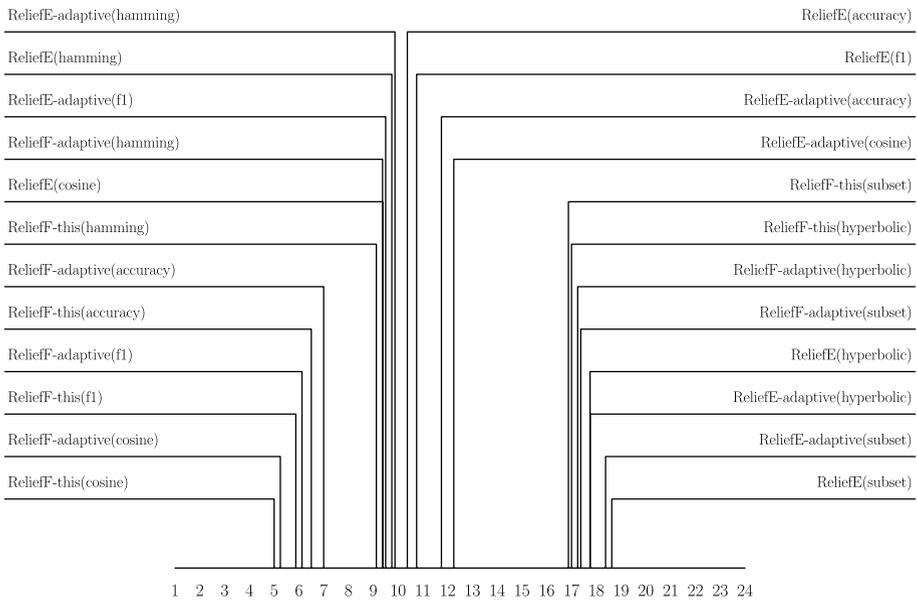
Fig. 26 Mean (upper) F1 scores for top 10 highly-ranked features. The average rank diagrams confirm the finding that if the target space is embedded via cosine distance, the MLC ReliefF variant performs the best
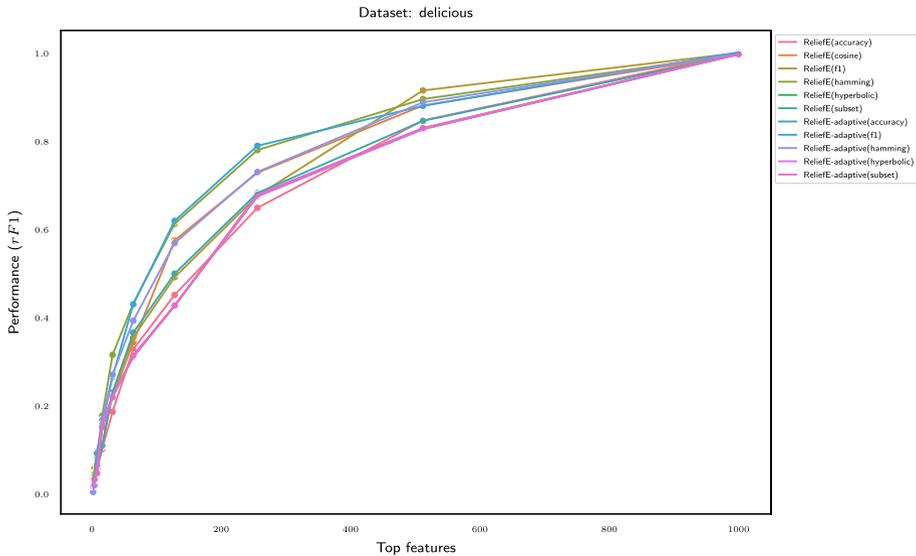
Fig. 27 Delicious performance curves. The ReliefE variants ReliefE-adaptive(f1) and ReliefE(hamming) perform consistently better up to 250 features

Appendix 9: Multi-label classification—case study with Delicious

We study in more detail the performance on the Delicious data set, as it offers interesting insights into the algorithms' performances (Fig. 27). The algorithms' performances are overall consistent. Note how cosine-based embeddings of the target space emerge as the best option (orange line), indicating embedding-based distances amongst the target instances can already offer competitive performance.

References

Alpaydin, E., & Kaynak, C. (1998). Cascading classifiers. *Kybernetika, 34*(4), 369–374.

Anguita, D., Ghio, A., Oneto, L., Parra, X., & Reyes-Ortiz, J. (2013). A public domain dataset for human activity recognition using smartphones. ESANN.

Armstrong, S. A., Staunton, J. E., Silverman, L. B., Pieters, R., den Boer, M. L., Minden, M. D., et al. (2002). Mll translocations specify a distinct gene expression profile that distinguishes a unique leukemia. *Nature Genetics, 30*(1), 41–47.

Arora, S., Hazan, E., & Kale, S. (2006). A fast random sampling algorithm for sparsifying matrices. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* (pp. 272–279). Springer.

Balasubramanian, M., & Schwartz, E. L. (2002). The isomap algorithm and topological stability. *Science, 295*(5552), 7–7.

Benavoli, A., Corani, G., Demšar, J., & Zaffalon, M. (2017). Time for a change: A tutorial for comparing multiple classifiers through Bayesian analysis. *The Journal of Machine Learning Research, 18*(1), 2653–2688.

Breskvar, M., Kocev, D., & Dzeroski, S. (2018). Ensembles for multi-target regression with random output selections. *Machine Learning, 107*(11), 1673–1709. https://doi.org/10.1007/s10994-018-5744-y.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine, 34*(4), 18–42.

Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D. M., Hill, A. J., et al. (2019). The single-cell transcriptional landscape of mammalian organogenesis. *Nature, 566*(7745), 496–502. https://doi.org/10.1038/s41586-019-0969-x.

Connor, J. T., Martin, R. D., & Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks, 5*(2), 240–254.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*(Jan), 1–30.

Dong, W., Moses, C., & Li, K. (2011). Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web* (pp. 577–586).

Džeroski, S., Blockeel, H., Kompare, B., Kramer, S., Pfahringer, B., & Van Laer, W. (1999). Experiments in predicting biodegradability. In textitInternational conference on inductive logic programming (pp. 80–91). Springer.

Eppstein, M. J., & Haake, P. (2008). Very large scale relieff for genome-wide association analysis. In *2008 IEEE symposium on computational intelligence in bioinformatics and computational biology* (pp. 112–119). IEEE.

Facco, E., d'Errico, M., Rodriguez, A., & Laio, A. (2017). Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports, 7*(1), 1–8.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems, 151,* 78–94.

Granizo-Mackenzie, D., & Moore, J. H. (2013). Multiple threshold spatially uniform relieff for the genetic analysis of complex human diseases. In *European conference on evolutionary computation, machine learning and data mining in bioinformatics* (pp. 1–10). Springer.

Greene, C. S., Himmelstein, D. S., Kiralis, J., & Moore, J. H. (2010). The informative extremes: Using both nearest and farthest individuals can improve relief algorithms in the domain of human genetics. In *European conference on evolutionary computation, machine learning and data mining in bioinformatics* (pp. 182–193). Springer.

Greene, C. S., Penrod, N. M., Kiralis, J., & Moore, J. H. (2009). Spatially uniform relieff (surf) for computationally-efficient filtering of gene-gene interactions. *BioData Mining, 2*(1), 5.

Guyon, I., Gunn, S., Ben-Hur, A., & Dror, G. (2005). Result analysis of the nips 2003 feature selection challenge. In *Advances in neural information processing systems* (pp. 545–552).

Han, E. H. S., & Karypis, G. (2000). Centroid-based document classification: Analysis and experimental results. In D. A. Zighed, J. Komorowski, & J. Żytkow (Eds.), *Principles of data mining and knowledge discovery* (pp. 424–431). Berlin: Springer.

Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory, 14*(1), 55–63. https://doi.org/10.1109/TIT.1968.1054102.

Imdb dataset. (2010). https://sourceforge.net/projects/meka/files/Datasets/IMDB-F.arff/download.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 discovery challenge*.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Kira, K., Rendell, L. A., et al. (1992). The feature selection problem: Traditional methods and a new algorithm. *AAAI, 2,* 129–134.

Lam, S.K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the LLVM compiler infrastructure in HPC* (pp. 1–6).

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Liu, T., Moore, A. W., Yang, K., & Gray, A. G. (2005). An investigation of practical approximate nearest neighbor algorithms. In *Advances in neural information processing systems* (pp. 825–832).

Maaten, L. .v.d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(Nov), 2579–2605.

Masci, J., Boscaini, D., Bronstein, M., & Vandergheynst, P. (2015). Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 37–45).

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

McInnes, L., Healy, J., Saul, N., & Grossberger, L. (2018). Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software, 3*(29), 861.

Mežnar, S., Lavrač, N., & Škrlj, B. (2020). Snore: Scalable unsupervised learning of symbolic node representations. *IEEE Access, 8,* 212568–212588. https://doi.org/10.1109/ACCESS.2020.3039541.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani & K.Q. Weinberger (Eds.), *Advances in neural information processing systems* (Vol. 26, pp. 3111–3119). Curran Associates, Inc.

Moore, J. H., & White, B. C. (2007). Tuning relieff for genome-wide genetic analysis. In *European conference on evolutionary computation, machine learning and data mining in bioinformatics* (pp. 166–175). Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*(Oct), 2825–2830.

Perovšek, M., Vavpetič, A., Kranjc, J., Cestnik, B., & Lavrač, N. (2015). Wordification: Propositionalization by unfolding relational data into bags of words. *Expert Systems with Applications, 42*(17–18), 6442–6456.

Pestian, J. P., Brew, C., Matykiewicz, P., Hovermale, D. J., Johnson, N., Bretonnel Cohen, K., & Duch, W. (2007). A shared task involving multi-label classification of clinical free text. In *Proceedings of the workshop on BioNLP 2007: Biological, translational, and clinical language processing (BioNLP'07)* (pp. 97–104).

Petković, M., Kocev, D., & Džeroski, S. (2018). Feature ranking with relief for multi-label classification: Does distance matter? In L. Soldatova, J. Vanschoren, G. Papadopoulos, & M. Ceci (Eds.), *Discovery science* (pp. 51–65). Cham: Springer International Publishing.

Petković, M., Slavkov, I., Kocev, D., & Džeroski, S. (2021). Biomarker discovery by feature ranking: Evaluation on a case study of embryonal tumors. *Computers in Biology and Medicine, 128,* 104143. https://doi.org/10.1016/j.compbiomed.2020.104143.

Petković, M., Škrlj, B., Kocev, D., & Simidjievski, N. (2020). Fuzzy Jaccard index: A robust comparison of ordered lists. https://arxiv.org/abs/2008.02216

Pomeroy, S., Tamayo, P., Gaasenbeek, M., Sturla, L. M., Angelo, M., McLaughlin, M., et al. (2002). Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature, 415,* 436–42. https://doi.org/10.1038/415436a.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., et al. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR), 51*(5), 1–36.

Robnik-Šikonja, M., & Kononenko, I. (2003). Theoretical and empirical analysis of relieff and rrelieff. *Machine Learning, 53*(1–2), 23–69.

Roweis, S. .T. ., & Saul, L. .K. . (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290*(5500), 2323–2326.

Sakar, B. E., Isenkul, M. E., Sakar, C. O., Sertbas, A., Gurgen, F., Delil, S., et al. (2013). Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics, 17*(4), 828–834.

Shapiro, A. D. (1984). The role of structured induction in expert systems. *Annexe Thesis Digitisation Project 2018 Block 19*.

Škrlj, B., Džeroski, S., Lavrač, N., & Petkovič, M. (2020). Feature importance estimation with self-attention networks. arXiv preprint arXiv:2002.04464.

Stepišnik, T., & Kocev, D. (2020). Hyperbolic embeddings for hierarchical multi-label classification. In D. Helic, G. Leitner, M. Stettinger, A. Felfernig, & Z. W. Raś (Eds.), *Foundations of intelligent systems* (pp. 66–76). Cham: Springer International Publishing.

Stiglic, G., & Kokol, P. (2010). Stability of ranked gene lists in large microarray analysis studies. *BioMed Research International, 2010*.

Stokes, M. E., & Visweswaran, S. (2012). Application of a spatially-weighted relief algorithm for ranking genetic predictors of disease. *BioData Mining, 5*(1), 20.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2008). Effective and efficient multilabel classification in domains with large number of labels. In *ECML/PKDD 2008 workshop on mining multidimensional data (MMD'08)*.

Ueda, N., & Saito, K. (2003). Parametric mixture models for multi-labeled text. In *Advances in neural information processing systems* (Vol. 15, pp. 721–728). MIT Press.

Urbanowicz, R. J., Olson, R. S., Schmitt, P., Meeker, M., & Moore, J. H. (2018). Benchmarking relief-based feature selection methods for bioinformatics data mining. *Journal of Biomedical Informatics, 85,* 168–188.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., et al. (2013). The cancer genome atlas pan-cancer analysis project. *Nature Genetics, 45*(10), 1113.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.