A novel Kernel Extension for the Nearest Feature Line Classifier

Hidden for double-blind review

***Abstract*—The Nearest Feature Line (NFL) rule represents a widely applied variant of the Nearest Neighbor rule whose usefulness has been shown in many different contexts. Even the NFL rule has been extended along different directions, only one of them is based on kernels: in this paper we make one step forward along this direction, proposing a novel kernel extension of the NFL rule, called *Kernel Rectified NFL* classifier. Our approach is based on tools and concepts coming from the Rectified Nearest Feature Line Segment (RNFLS) classifier, another extension of the NFL rule which is aimed at solving known problems of the original one, such as extrapolation and interpolation inaccuracies. In the paper we present the method, together with an empirical evaluation based on 15 different datasets, which show that kernelization is indeed a promising direction for extending the NFL family of classifiers.**

***Index Terms*—Nearest Feature Line, Rectified Nearest Feature Line Segment, Kernels**

I. Introduction

The well-known nearest neighbor (1-NN) classifier [1] assigns an unseen object to the class of the less dissimilar object found within a set of prototype examples and according to a given distance measure. Many enhancements of 1-NN have been proposed in the literature; among them, the most popular variants consist on either reductions or enlargements of the set of prototypes [2], while others are based on using problem-specific dissimilarity measures [3] or weights [4], [5]. A seminal extension of 1-NN that has attracted attention during the last decades is the so-called Nearest Feature Line (NFL) rule [6] which, in turn, inspired the proposition of a whole family of methods originally termed by Chien and Wu [7] as the Nearest Feature Classifiers. NFL, in brief, consists in building a line connecting each pair of prototype points belonging to the same class, such that the class label assignment for a test object is now based on the label of the nearest line instead of deciding it according to the class of the nearest prototype point. NFL performs well in high-dimensional spaces, but suffers from intrinsic weaknesses in low-dimensional ones. Many extensions and solutions have been proposed to cope with the NFL weaknesses, which exploit different ideas like the selection of feature lines [8]–[11], the transformation of the original data or the consideration of alternative representation spaces [12], [13] as well as the design of better point-to-line distance measures [14]–[16].

However —to the best of our knowledge— there is only one extension of NFL, proposed in [17], which exploits *kernel tools* [18]; this is somehow surprising, since many different studies in the Pattern Recognition and Machine Learning community proposed and showed the usefulness of kernelized extensions of existing methods, not just by using the well-known support vector machines to solve classification problems, but also in related tasks such as feature extraction, dimensionality reduction and clustering. Among the two latter, the proposal of Kernel Principal Component Analysis [19], [20] and Kernel Kmeans [21], [22] stand out as reference kernel techniques in their respective areas. The idea behind the many different works on kernelized approaches consists in re-writing the procedure in terms of dot products, such that the so-called kernel trick can be applied to implicitly project points in a high dimensional space, without changing the procedure, where a number of advantageous behaviors are obtained. In spite of the advent of deep learning —and, particularly Convolutional Neural Networks [23]— as the current dominant trend, kernel methods continue to be effective solutions in both supervised and unsupervised learning.

In this paper we make a step forward along this direction, by proposing a novel kernel extension of the NFL classifier. We start from the observation that the kernelized version proposed in [17] is related to the original NFL scheme, and does not exploit all the advances of NFL that have been presented in more recent years. To cope with this limitation, here we propose the *Kernel Rectified NFL* classifier, an advanced kernelized NFL classifier based on ideas and concepts coming from the Rectified Nearest Feature Line Segment (RNFLS) classifier [9], [24]. This version was proposed as a solution for two diagnosed problems of NFL: extrapolation and interpolation inaccuracies. The former inaccuracy is due to the fact that feature lines extend indefinitely beyond their endpoints in both directions: this represents a serious problem in low-dimensional spaces, but it is of limited harm in high-dimensional ones, as observed by [9]. The latter inaccuracy is due to the potential invasion of the interpolating segment of the feature lines –typically for multimodal classes– to the territory of other classes, thus more related to the complexity of the problem. If we consider the kernelization perspective, we can say that extrapolation may represent a less crucial issue, since kernelization permits to work in high (possibly infinite) dimensional spaces; on the contrary, the risk of interpolation inaccuracy is still there, since classes can still be multimodal. For these reasons, our extension exploits only the rectification process, which is aimed at reducing the interpolation problem by removing lines which “cross” the territory of other classes.

Our formulation of the *Kernel Rectified NFL* classifier starts from the NFL version that is purely based on dissimilarities [13], and proposes a kernelized variant of both the classifier and the rectification process. Experimental evaluations, conducted on 15 different datasets, are promising, showing that our Kernel Rectified NFL approach very often outperforms the alternatives Moreover, our results provide a clear evidence that kernelization is very beneficial for these types of methods, including the original 1-NN and NFL classifiers, encouraging us to go ahead along this line of research.

The remaining part of the paper is organized as follows. The background concepts on the NFL rule, its rectified version and about kernels methods are provided in Sec. II. A detailed explanation of our proposal is presented in Sec. III. Experimental results are shown and discussed in Sec. IV. Finally, our concluding remarks and recommendations are given in Sec. V.

II. Background

A. The Nearest Feature Line (NFL) and the Rectified Nearest Feature Line Segment (RNFLS)

The Nearest Feature Line (NFL) classifier was originally proposed in [6] as an extension of the well known Nearest Neighbor rule, which, given a set of n labeled prototype examples $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ and a test point $\boldsymbol{x}$ to be classified, assigns $\boldsymbol{x}$ with the class label $y_{i'}$, such that

$$i' = \underset{i=1,\ldots,n}{\arg\min}\, d(\boldsymbol{x}, \boldsymbol{x}_i) \triangleq \underset{i=1,\ldots,n}{\arg\min} ||\boldsymbol{x} - \boldsymbol{x}_i||. \quad (1)$$

NFL can be simply understood as the very same rule from Eq. (1) but using now a larger collection of prototypes built from the original set $\{(\boldsymbol{x}_i, y_i)\}$, such as lines.

More precisely, let $\overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ a line in the feature space, having a class label $\overset{\leftrightarrow}{y}_k$ and connecting two feature points $\boldsymbol{x}_i, \boldsymbol{x}_j$ provided that $y_i = y_j$ and $i \neq j$. Therefore, $\overset{\leftrightarrow}{y}_k = y_i = y_j$. Moreover, let n_ℓ be the number of feature lines that can be built under these constraints. Then, the NFL rule assigns $\boldsymbol{x}$ with the class label $\overset{\leftrightarrow}{y}_{k'}$, such that

$$k' = \underset{k=1,\ldots,n_\ell}{\arg\min}\, d\left(\boldsymbol{x}, \overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), \quad (2)$$

where

$$d\left(\boldsymbol{x}, \overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) \triangleq ||\boldsymbol{x} - [(1-\mu)\boldsymbol{x}_i + \mu \boldsymbol{x}_j]\,|| \quad (3)$$

and

$$\mu = \frac{(\boldsymbol{x} - \boldsymbol{x}_i) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_i)}{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}. \quad (4)$$

Instead of a single $\boldsymbol{x}$ consider three different test points —$\boldsymbol{x}_a$, $\boldsymbol{x}_b$ and $\boldsymbol{x}_c$— whose projections onto $\overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ lie on the left-extrapolating part, the interpolating part and the right-extrapolating part of the line, respectively. A graphical illustration of the NFL mechanism for those three different situations is shown in Fig. 1.

NFL works very well in high-dimensional spaces; however, feature lines tend to cross each other in lower-dimensional ones and, consequently, NFL may not perform well in such dimensions. One extension which is aimed at improving NFL is the the Rectified Nearest Feature Line Segment (RNFLS) procedure. This method was originally proposed in [9] as a solution for two diagnosed problems of NFL: extrapolation and interpolation inaccuracies. The former problem arises because feature lines extend indefinitely beyond their endpoints in both directions, and is solved by a process called *segmentation*. The second problem is due to potential invasion of the interpolating segment of the feature lines –typically for multimodal classes– to the territory of other classes, and is solved by a process called *rectification*.

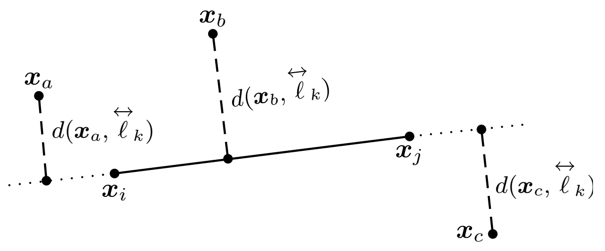
$\boldsymbol{x}_b$
$\boldsymbol{x}_a$
$d(\boldsymbol{x}_b, \overset{\leftrightarrow}{\ell}_k)$
$d(\boldsymbol{x}_a, \overset{\leftrightarrow}{\ell}_k)$
$\boldsymbol{x}_j$
$\boldsymbol{x}_i$
$d(\boldsymbol{x}_c, \overset{\leftrightarrow}{\ell}_k)$
$\boldsymbol{x}_c$

Fig. 1. An illustration of the NFL decision rule for three different test cases.

More in details, segmentation consists in not considering the whole (uncut) feature line $\overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ but, instead, a so-called feature line segment $\overset{\mapsto}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)$; see Fig. 2. In practice, the distance from a test point to a feature line segment is obtained as follows:

$$d\left(\boldsymbol{x}, \overset{\mapsto}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right) = \begin{cases} d\left(\boldsymbol{x}, \overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)\right), & \text{if } 0 < \mu < 1; \\ d(\boldsymbol{x}, \boldsymbol{x}_i), & \text{if } \mu < 0; \\ d(\boldsymbol{x}, \boldsymbol{x}_j), & \text{if } \mu > 1. \end{cases} \quad (5)$$

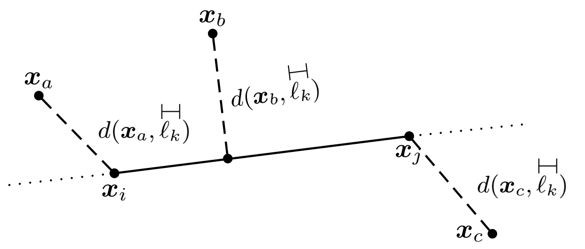
$\boldsymbol{x}_b$
$\boldsymbol{x}_a$
$d(\boldsymbol{x}_b, \overset{\mapsto}{\ell}_k)$
$d(\boldsymbol{x}_a, \overset{\mapsto}{\ell}_k)$
$\boldsymbol{x}_j$
$\boldsymbol{x}_i$
$d(\boldsymbol{x}_c, \overset{\mapsto}{\ell}_k)$
$\boldsymbol{x}_c$

Fig. 2. Illustration of the segmentation process in RNFLS.

Rectification is a selection of those feature line segments that do not cross the territory of other class, according to the following criterion evaluated for each prototype $\boldsymbol{x}_m$, $m = 1, \ldots, n$ having an associated territory with radius $r_m = \min_{\forall i, y_i \neq y_m} d(\boldsymbol{x}_i, \boldsymbol{x}_m)$:

$$\overset{\mapsto}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ is preserved if } d(x_m, \overset{\mapsto}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j)) > r_m, \forall m$$
$$\text{and } y_m \neq \overset{\mapsto}{y}_k; \text{otherwise, } \overset{\mapsto}{\ell}_k(\boldsymbol{x}_i, \boldsymbol{x}_j) \text{ is excluded.} \quad (6)$$

Let n_s be the number of selected feature line segments after rectification. Since only some of the feature lines segments are selected by this process, then $n_s \leq n_\ell$.

B. Kernel Methods

The effectiveness of kernel methods to solve complicated problems lies on their ability to map the data to a higher dimensional space, where non-linear problems can be cast into linear ones. The mapping can be implicitly done via the so-called kernel trick [25, Chap. 12]. Let $\boldsymbol{x}_a$, $\boldsymbol{x}_b$ be two feature vectors in $\mathbb{R}^p$ and $\Phi(\cdot)$ a mapping function from $\mathbb{R}^p$ to $\mathbb{R}^q$, where $q > p$. The dot product between the mapped points can be implicitly computed via a specific function, called *Kernel*: $K(\boldsymbol{x}_a, \boldsymbol{x}_b) = \langle \Phi(\boldsymbol{x}_a), \Phi(\boldsymbol{x}_b) \rangle$. In words, the trick consists in the ability of computing the dot product between the mapped feature vectors in terms of the evaluation of a function (the kernel) whose arguments are the vectors in their original dimension. The most commonly used kernels are the polynomial kernel and the radial basis function (rbf) kernel. The latter is particularly attractive because, with it, the resulting high dimensional feature space can be even infinite-dimensional.

III. The Proposed Method: Ker-RNFL

In this section the proposed approach is presented. In particular we propose a novel kernelized NFL classifier, which exploits kernelization of the basic NFL classifier to which we add the Rectification process of the RNFLS classifier, leading to the *Kernel Rectified NFL* classifier: abbreviated as *Ker-RNFL*. Please note that we add only the rectification process, and not the segmentation step. Actually, as explained in Sec. I, segmentation aims at solving the problem of extrapolation, but as observed by [9], this represents a serious problem only in low-dimensional spaces. Since kernelization would permit to work in a higher dimensional space, we reduced our attention to the Rectification process, which is aimed at facing problems more related to the true complexity of the problem (classes overlap).

In order to derive a kernelized version of an algorithm, the classical approach is to rewrite such procedure only in terms of dot products [25]: by replacing dot products with kernels we can have the procedure working in a high dimensional space, where the task may be simpler. Our kernelization strategy is slightly different, and is based on two steps.

- **Step 1**. In the first step we observe that the NFL and the Rectification procedures can be written only in terms of pairwise distances between objects. Actually, both steps need to compute the distance between an object and a line (the rectification also needs to compute the radius, which is already defined in terms of distances). The object-line distance can be written only in terms of pairwise distances between objects by exploiting the formulation proposed in [13], where a version of NFL was proposed able to work in cases when only pairwise dissimilarities between objects are available but not the point coordinates in a feature space. For that, the Heron's formula is used along with the typical way to find the height of a triangle, as follows:
$$d(x, \overset{\leftrightarrow}{\ell}_k(x_i, x_j)) = \frac{2\sqrt{s(s - d(x_j, x))(s - d(x_i, x_j))(s - d(x_i, x))}}{d(x_i, x_j)}, \qquad (7)$$
where $s = (d(x_j, x) + d(x_i, x_j) + d(x_i, x))/2$. This formulation was introduced so that it can work only with dissimilarities, which might be derived directly from the objects themselves, without the need of having their associated coordinates in a vector space.
- **Step 2**. Given the version with only pairwise distances, a kernelized version can be obtained by exploiting the relation between the (squared) Euclidean distance between two vectors $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and the dot product [25]:
$$||\boldsymbol{x}_i - \boldsymbol{x}_j||^2 = \boldsymbol{x}_i \cdot \boldsymbol{x}_i + \boldsymbol{x}_j \cdot \boldsymbol{x}_j - 2(\boldsymbol{x}_i \cdot \boldsymbol{x}_j). \qquad (8)$$

If we want to compute the kernelized version, we have to replace all pairwise distances $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in our formulation with its kernelized version $d^K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e.

$$d^K(\boldsymbol{x}_i, \boldsymbol{x}_j) = K(\boldsymbol{x}_i, \boldsymbol{x}_i) + K(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2K(\boldsymbol{x}_i, \boldsymbol{x}_j). \qquad (9)$$

Depending on the chosen kernel, we can arrive to different versions. For example, if we use the rbf kernel function [25]:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/\sigma}$$

then it is easy to see that the distance in (7) can be written as:

$$d^K(x, \overset{\leftrightarrow}{\ell}_k(x_i, x_j)) = \frac{\sqrt{s^K A(\boldsymbol{x}_j, \boldsymbol{x}) A(\boldsymbol{x}_i, \boldsymbol{x}_j) A(\boldsymbol{x}_i, \boldsymbol{x})}}{1 - K(\boldsymbol{x}_i, \boldsymbol{x}_j)}, \qquad (10)$$

where

$$\begin{aligned} A(\boldsymbol{x}_j, \boldsymbol{x}) &= 1 - K(\boldsymbol{x}_i, \boldsymbol{x}_j) - K(\boldsymbol{x}_i, \boldsymbol{x}) + K(\boldsymbol{x}, \boldsymbol{x}_j) \\ A(\boldsymbol{x}_i, \boldsymbol{x}_j) &= 1 - K(\boldsymbol{x}, \boldsymbol{x}_j) - K(\boldsymbol{x}, \boldsymbol{x}_i) + K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ A(\boldsymbol{x}_i, \boldsymbol{x}) &= 1 - K(\boldsymbol{x}, \boldsymbol{x}_j) - K(\boldsymbol{x}_i, \boldsymbol{x}_j) + K(\boldsymbol{x}, \boldsymbol{x}_i) \end{aligned}$$

and

$$s^K = 3 - K(\boldsymbol{x}_i, \boldsymbol{x}_j) - K(\boldsymbol{x}_i, \boldsymbol{x}) - K(\boldsymbol{x}, \boldsymbol{x}_j)$$

In the same way, the kernelized radius r^K_m for the rectification process can be computed as:

$$r^K_m = \min_{\forall i, y_i \neq y_m} (1 - K(\boldsymbol{x}_i, \boldsymbol{x}_m)) = \max_{\forall i, y_i \neq y_m} K(\boldsymbol{x}_i, \boldsymbol{x}_m) \qquad (11)$$

Summarizing, given these definitions, the proposed Kernel Rectified NFL works as follows:

- perform the rectification process, i.e. keep only those kernel lines $\overset{\vdash\!\dashv K}{\ell}_k (\boldsymbol{x}_i, \boldsymbol{x}_j)$ for which
$$d^K(x_m, \overset{\vdash\!\dashv K}{\ell}_k (\boldsymbol{x}_i, \boldsymbol{x}_j)) > r^K_m, \forall m \text{ and } y_m \neq \overset{\vdash\!\dashv K}{y}_k \qquad (12)$$
where $d^K(\cdot, \cdot)$ and r^K_m are computed using Eqs. (10) and (11), respectively.

- assigns a testing point $\boldsymbol{x}$ with the class label $\overset{\leftrightarrow}{y}_{k'}$, such that

$$k' = \underset{k=1,\ldots,n_\ell}{\arg\min}\, d^K\left(\boldsymbol{x}, \overset{\leftrightarrow}{\ell}_k(\boldsymbol{x}_i,\boldsymbol{x}_j)\right), \tag{13}$$

where again $d^K(\cdot,\cdot)$ is computed using Eq. (10).

A final note: even if having the same goal, the kernelization of NFL presented in [17] followed a different perspective, not exploiting the distance-based formulation of NFL. Moreover, in our derivation, we exploit it and extend the kernelization to the RNFLS variant, which was completely missing from the literature. As it will be demonstrated in the subsequent section, taking advantage of both kernelization and rectification, allows to simultaneously inherit the benefits of both processes.

IV. Experimental results

A heterogeneous collection of datasets (see Table I) was used for the experiments in order to evaluate, under diverse circumstances, the performance of the proposed method with respect to some related methods. In particular we grouped our evaluated methods in two families:

- *Original versions*: here we considered all non-kernel versions. In particular we analysed the standard Nearest Neighbor rule ('NN' in the tables), the original Nearest Feature Line ("NFL"), and its extension: the Rectified Nearest Feature Segment Line ("RNFLS");
- *Kernelized versions*: here we considered the kernelized Nearest Neighbor rule ("Ker-NN"), as proposed in [26], which represents a kernelized version of the NN rule, the Kernel NFL ("Ker-NFL"), as proposed in [17], and the proposed approach ("Ker-RNFL").

Please note that with this comparison we can also evaluate how beneficial is the kernelization procedure for the different rules. Most of the datasets are publicly available at the UCI Machine Learning Repository, except for the Ruiz200TriaxalOnly that comes from a Colombian Seismological Observatory, and deals with the classification of volcano-seismic signals – for more info please see [27], and the Flickr dataset, which represents the first two classes of the Flickr dataset available at [28].

TABLE I
Datasets considered in the empirical evaluation of Ker-RNFL

Dataset	Obj	Dims	Classes
Iris	150	4	3
Lung	32	54	3
Glass	214	9	4
BreastTissue	106	9	6
Wine	178	13	3
Heart	297	13	2
Fertility	100	9	2
Parkinsons	195	22	2
Sonar	208	60	2
ColonCancer	62	2000	2
Ruiz200TriaxalOnlyZ	200	26	2
Flickr	600	82	2
Energy	1500	24	50
Ionosphere	351	34	2
WBC	683	9	2

In all the experiments, the Euclidean distance was used as the by-default nearness criterion. Classification errors with Averaged Holdout Cross Validation (for 30 repetitions of the experiments) were computed. For each repetition, the data was normalized with respect to the variance of the training partitions; that is, by using z-score standardization. In the case of the kernelized methods, several parameterizations for both polynomial and rbf kernels were explored; namely using polynomial kernels with degrees in $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$ and rbf kernels with σ parameter in $\{0.01, 0.02, 0.05, 0.075, 0.1, 0.2, 0.5, 0.75, 1, 2, 5, 7.5, 10, 20, 50, 75, 100\}$, respectively.

Due to the high computational complexity of the NFL family of classifiers (the number of lines grows quadratically with respect to the number of points), we also investigate the behaviour of such methods when using only a random fraction of all lines. In particular we evaluated the results for 50%, 70% and 100% (the whole set of lines). For the kernel methods, we only select the best kernel for every cross-validation run in every problem. Results are reported in the left part of Tables II, III and IV. The best results per dataset are highlighted in light gray in the three tables. In addition, in the right part of the tables, the result of paired t-tests (confidence level of 0.05) comparing the original versions of the methods against their kernelized counterparts are also shown. The result in the column indicates which of the two variants (Kernelized or Original) is better with a statistically significant difference.

The first general observation is that, in all the tables except for one dataset in Table II, Ker-NFL and Ker-RNFL are the best performing classifiers. In addition, it is interesting to highlight that, as the fraction of lines used increases, the number of times when Ker-RNFL is better than Ker-NFL slightly increase as well. Notice that, in Table II, both Ker-NFL and Ker-RNFL are the best performing classifiers in 7 out of 15 occasions. Then, when the fraction of lines increases from 50% to 70%, the proportion is 9 times vs. 6 times in favor of Ker-RNFL; see Table III. Finally, when considering the whole set of lines (Table IV) the proportion is 10 times vs. 5 times also in favor of the proposed method.

When comparing in more detail the classification errors of Ker-RNFL vs. those of Ker-NFL, it is worth to highlight that, for the seven cases where Ker-RNFL consistently outperformed Ker-NFL independently of the fractions of lines used, the largest differences in favor of the proposed method correspond to the Heart, Ruiz200TriaxalOnlyZ and Flick datasets. The common characteristic shared by these datasets seem to be a relatively large number of objects per class (i.e. a relatively large Obj/Classes ratio) in a moderate-to-large dimensionality of the original representational space. For such three datasets, the Obj/Classes ratios are 148.5, 100 and 300, respectively, in original dimensionalities of 13, 26 and 82 dimensions. This characteristic was also observed for Ionosphere; however, for this dataset, Ker-RNFL was better than Ker-NFL only when using the whole fraction of lines. In spite of that, the difference between the performances of Ker-RNFL and Ker-NFL for that dataset is not that much.

TABLE II
AVERAGED CLASSIFICATION ERRORS. FRACTION OF LINES USED: 0.5

	Original versions			Kernelized versions			t-tests		
Dataset	NN	NFL	RNFLS	Ker-NN	Ker-NFL	Ker-RNFL	NN vs KerNN	NFL vs Ker-NFL	RNFLS vs Ker-RNFL
Iris	0.0591	0.1031	0.0427	0.0493	0.0409	0.0276	Kernel	Kernel	Kernel
Lung	0.5356	0.5111	0.5222	0.4867	0.4422	0.4356	Kernel	Kernel	Kernel
Glass	0.2978	0.3522	0.2701	0.2855	0.2717	0.2717	Kernel	Kernel	Equal
BreastTissue	0.3474	0.3814	0.3571	0.3333	0.3071	0.3462	Kernel	Kernel	Equal
Wine	0.0519	0.0447	0.0318	0.0443	0.0258	0.0227	Kernel	Kernel	Kernel
Heart	0.2338	0.2176	0.1696	0.2297	0.2083	0.1572	Kernel	Kernel	Kernel
Fertility	0.1807	0.142	0.1187	0.112	0.11	0.1127	Kernel	Kernel	Equal
Parkinsons	0.0931	0.0921	0.1158	0.0763	0.067	0.0787	Kernel	Kernel	Kernel
Sonar	0.1602	0.1547	0.166	0.1602	0.1337	0.1469	Equal	Kernel	Kernel
ColonCancer	0.2753	0.271	0.3043	0.2516	0.2097	0.2323	Kernel	Kernel	Kernel
Ruiz200TriaxalOnlyZ	0.3553	0.3703	0.3237	0.3223	0.302	0.2833	Kernel	Kernel	Kernel
Flickr	0.3606	0.3308	0.2798	0.3269	0.3102	0.2743	Kernel	Kernel	Kernel
Energy	0.0577	0.055	0.0618	0.0577	0.0499	0.05	Equal	Kernel	Kernel
Ionosphere	0.156	0.1693	0.1093	0.0709	0.0684	0.0699	Kernel	Kernel	Kernel
WBC	0.0429	0.0495	0.0262	0.0395	0.0311	0.0252	Kernel	Kernel	Equal

TABLE III
AVERAGED CLASSIFICATION ERRORS. FRACTION OF LINES USED: 0.7

	Original versions			Kernelized versions			t-tests		
Dataset	NN	NFL	RNFLS	Ker-NN	Ker-NFL	Ker-RNFL	NN vs KerNN	NFL vs Ker-NFL	RNFLS vs Ker-RNFL
Iris	0.0631	0.1018	0.0409	0.0502	0.0351	0.0236	Kernel	Kernel	Kernel
Lung	0.5467	0.5	0.4822	0.5111	0.4356	0.46	Kernel	Kernel	Equal
Glass	0.2975	0.3487	0.2638	0.2874	0.2673	0.2591	Kernel	Kernel	Equal
BreastTissue	0.334	0.4045	0.3564	0.316	0.3013	0.3385	Kernel	Kernel	Equal
Wine	0.0492	0.0432	0.0261	0.0432	0.0239	0.0205	Kernel	Kernel	Kernel
Heart	0.2419	0.225	0.168	0.2331	0.2088	0.1491	Kernel	Kernel	Kernel
Fertility	0.1787	0.154	0.12	0.116	0.12	0.1133	Kernel	Kernel	Kernel
Parkinsons	0.0749	0.0935	0.0997	0.067	0.0546	0.0632	Kernel	Kernel	Kernel
Sonar	0.1731	0.1498	0.1602	0.1728	0.1311	0.1466	Equal	Kernel	Kernel
ColonCancer	0.2828	0.2624	0.3	0.257	0.1957	0.2237	Kernel	Kernel	Kernel
Ruiz200TriaxalOnlyZ	0.369	0.379	0.317	0.3383	0.321	0.2693	Kernel	Kernel	Kernel
Flickr	0.365	0.3368	0.2824	0.3263	0.311	0.2786	Kernel	Kernel	Equal
Energy	0.0587	0.0543	0.0619	0.0586	0.0502	0.051	Equal	Kernel	Kernel
Ionosphere	0.1562	0.1621	0.101	0.0695	0.0659	0.0682	Kernel	Kernel	Kernel
WBC	0.0438	0.0514	0.0271	0.0388	0.0305	0.0254	Kernel	Kernel	Equal

TABLE IV
AVERAGED CLASSIFICATION ERRORS. FRACTION OF LINES USED: 1

	Original versions			Kernelized versions			t-tests		
Dataset	NN	NFL	RNFLS	Ker-NN	Ker-NFL	Ker-RNFL	NN vs KerNN	NFL vs Ker-NFL	RNFLS vs Ker-RNFL
Iris	0.0596	0.1071	0.0378	0.0502	0.0378	0.0213	Kernel	Kernel	Kernel
Lung	0.5222	0.4756	0.4778	0.5044	0.4378	0.4244	Kernel	Kernel	Kernel
Glass	0.295	0.3443	0.2689	0.2865	0.2711	0.2538	Kernel	Kernel	Kernel
BreastTissue	0.3372	0.4179	0.3263	0.3244	0.3019	0.3237	Kernel	Kernel	Equal
Wine	0.0481	0.0489	0.0208	0.0443	0.0216	0.0178	Kernel	Kernel	Equal
Heart	0.2333	0.2214	0.1784	0.2259	0.2041	0.157	Kernel	Kernel	Kernel
Fertility	0.1693	0.1507	0.1187	0.1113	0.112	0.1093	Kernel	Kernel	Kernel
Parkinsons	0.0749	0.0921	0.0997	0.0632	0.056	0.0643	Kernel	Kernel	Kernel
Sonar	0.1722	0.1576	0.1625	0.1702	0.1392	0.1479	Equal	Kernel	Kernel
ColonCancer	0.3	0.2731	0.2903	0.2591	0.2086	0.243	Kernel	Kernel	Kernel
Ruiz200TriaxalOnlyZ	0.3583	0.3777	0.3283	0.329	0.313	0.281	Kernel	Kernel	Kernel
Flickr	0.3586	0.3382	0.2816	0.3261	0.3081	0.2766	Kernel	Kernel	Kernel
Energy	0.056	0.0527	0.0556	0.056	0.0477	0.0481	Equal	Kernel	Kernel
Ionosphere	0.1455	0.1537	0.0907	0.0636	0.0623	0.0619	Kernel	Kernel	Kernel
WBC	0.0453	0.048	0.0271	0.0375	0.0299	0.0237	Kernel	Kernel	Kernel

As another observation, it can be seen that in almost all cases the kernelized version of a given rule outperforms its original counterpart: this represents a promising result, encouraging us to go ahead along the direction of kernelizing the different extensions of the NFL classifier.

V. Conclusion

A novel kernel extension of the NFL rule, called *Ker-RNFL* classifier was proposed in this paper. The approach successfully inherit not just the good representational capabilities of NFL but also the corrections of its weaknesses via the so-called rectification process and the well-known properties provided by kernel methods. The proposal, as well as the kernelized versions of the baseline approaches, were empirically evaluated on 15 different datasets. Results of Ker-RNFL vs. Ker-NFL show that the first approach tend to be better than the second one as the fraction of considered lines increases. The benefit of the proposed approach is specially noteworthy for datasets with a relatively large ratio of number of objects to number of classes, in moderate-to-large original dimensionalities. Such a behavior suggests that the combined contributions of the kernel trick and the rectified feature lines are profitable when point clouds per class, in the original space, exhibit a moderate tradeoff between compactness and sparseness.

Finally, the overall good results that were observed for the finely-tuned (best a posteriori solution) versions of the kernelized methods confirm the well-known fact that selecting appropriate kernel parameters is crucial in kernel methods. However, the aim of this paper was only showing the potential of the Ker-RNFL method, leaving as future work the problem of a proper kernel parameter selection.

Acknowledgment

Hidden for double-blind review.

References

[1] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[2] M. Lozano, J. M. Sotoca, J. S. Sánchez, F. Pla, E. Pekalska, and R. P. W. Duin, "Experimental study on prototype optimisation algorithms for prototype-based classification in vector spaces," *Pattern Recognition*, vol. 39, no. 10, pp. 1827–1838, 2006.

[3] E. Pękalska and R. P. W. Duin, "Dissimilarity measures," in *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, ser. Machine Perception and Artificial Intelligence. Singapore: World Scientific, 2005, vol. 64, ch. 5, pp. 215–254.

[4] S. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, 1976.

[5] M. Bicego and M. Loog, "Weighted k-nearest neighbor revisited," in *23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 1642–1647.

[6] S. Z. Li and J. Lu, "Face recognition using the nearest feature line method," *IEEE Transactions on Neural Networks*, vol. 10, no. 2, pp. 439–443, 1999.

[7] J.-T. Chien and C.-C. Wu, "Discriminant waveletfaces and nearest feature classifiers for face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 12, pp. 1644–1649, 2002.

[8] Q.-B. Gao and Z.-Z. Wang, "Center-based nearest neighbor classifier," *Pattern Recognition*, vol. 40, no. 1, pp. 346 – 349, 2007.

[9] H. Du and Y. Q. Chen, "Rectified nearest feature line segment for pattern classification," *Pattern Recognition*, vol. 40, no. 5, pp. 1486 – 1497, 2007.

[10] D.-Q. Han, C.-Z. Han, and Y. Yang, "A novel classifier based on shortest feature line segment," *Pattern Recognition Letters*, vol. 32, no. 3, pp. 485 – 493, 2011.

[11] K. Kamaei and H. Altınçay, "Editing the nearest feature line classifier," *Intelligent Data Analysis*, vol. 19, no. 3, pp. 563–580, 2015.

[12] H. Altınçay and Z. Erenel, "Avoiding the interpolation inaccuracy in nearest feature line classifier by spectral feature analysis," *Pattern Recognition Letters*, vol. 34, no. 12, pp. 1372–1380, 2013.

[13] M. Orozco-Alzate, R. P. W. Duin, and C. G. Castellanos-Domínguez, "A generalization of dissimilarity representations using feature lines and feature planes," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 242–254, feb 2009.

[14] W. Li, Q. Ruan, and J. Wan, "Graph-preserving shortest feature line segment for dimensionality reduction," *Neurocomputing*, vol. 110, pp. 80–91, jun 2013.

[15] Q. Feng, J.-S. Pan, and T.-S. Pan, "Feature curve metric for image classification," in *Modern Advances in Applied Intelligence*, M. Ali, J.-S. Pan, S.-M. Chen, and M.-F. Horng, Eds. Cham: Springer International Publishing, 2014, pp. 263–272.

[16] J.-S. Pan, Q. Feng, L. Yan, and J.-F. Yang, "Neighborhood feature line segment for image classification," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 387–398, 2015.

[17] Y. He, "Face recognition using kernel nearest feature classifiers," in *2006 International Conference on Computational Intelligence and Security*, vol. 1, nov 2006, pp. 678–683.

[18] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, dec 2002.

[19] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[20] ——, "Kernel principal component analysis," in *Advances in Kernel Methods—Support Vector Learning*. Cambridge, MA: MIT Press, 1999, pp. 327–352.

[21] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 780–784, 2002.

[22] I. S. Dhillon, Y. Guan, and B. Kulis, "Kernel K-Means: Spectral Clustering and Normalized Cuts," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2004, p. 551556.

[23] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[24] M. Orozco-Alzate and M. Bicego, "A cheaper Rectified-Nearest-Feature-Line-Segment classifier based on safe points," in *Proc. of the 25th Int. Conf. on Pattern Recognition (ICPR 2020)*. IEEE Computer Society, jan 2021, pp. 2787–2794.

[25] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[26] K. Yu, L. Ji, and X. Zhang, "Kernel nearest-neighbor algorithm," *Neural Processing Letters*, vol. 15, no. 2, pp. 147–156, 2002.

[27] M. Orozco-Alzate, P. Castro-Cabrera, M. Bicego, and J. Londoño-Bonilla, "The DTW-based representation space for seismic pattern classification," *Computers & Geosciences*, vol. 85, pp. 86–95, 2015.

[28] C. Segalin, "A new soft biometric trait: Favorite images." [Online]. Available: http://www.cristinasegalin.com