FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 49 (2024) No. 3
ISSN 0867-6356
DOI: 10.2478/fcds-2024-0015 e-ISSN 2300-3405

On feature extraction using distances from reference points

Maciej Piernik, Tadeusz Morzy, Robert Susmaga, Izabela Szczęch *

Abstract. Feature extraction is the key to a successfully trained classifier. Although many automatic methods exist for traditional data, other data types (e.g., sequences, graphs) usually require dedicated approaches. In this paper, we study a universal feature extraction method based on distance from reference points. First, we formalize this process and provide an instantiation based on network centrality. To reliably select the best reference points, we introduce the notion of θ-neighborhood which allows us to navigate the topography of fully connected graphs. Our experiments show that the proposed peak selection method is significantly better than a traditional top-k approach for centrality-based reference points and that the quality of the reference points is much less important than their quantity. Finally, we provide an alternative, neural network interpretation of reference points, which paves a path to optimization-based selection methods, together with a new type of neuron, called the Euclidean neuron, and the necessary modifications to backpropagation.

Keywords: Feature extraction, Classification, Reference points

1. Introduction

Classification of arbitrary objects without explicitly defined features requires one of two approaches: distance-based classification or feature extraction. The main advantage of the first approach is that it is able to operate on the full information available in the objects (given the right distance measure), however, is very limiting in terms of the available classifiers — basically to k-NN and Similarity Forests [18]. The second, more common approach does not have this limitation, however, requires a dedicated feature extraction method for each specific type of objects being processed. After performing feature extraction, a model can be trained using any standard classifier.

In this paper, we discuss a third possibility — one which combines the generality of the first group with the flexibility of the second group — namely, a universal

*Institute of Computing Science, Poznan University of Technology, Piotrowo 2, 60–965 Poznan, Poland, {Maciej.Piernik, Tadeusz.Morzy, Robert.Susmaga, Izabela.Szczech}@cs.put.poznan.pl

distance-based feature extraction method using reference points. We propose a novel reference points selection method based on network centrality and compare it against two baselines: random and clustering-based method, to identify which one produces the best features from the point of view of classification. In order to better select the top points according to network centrality, we put forward a new definition of neighborhood, called θ-neighborhood, capable of navigating the topography of objects based only on the distances between them, without the underlying space. We experimentally evaluate our proposals on 15 publicly available datasets with appropriate statistical analysis to draw reliable conclusions. Finally, we provide a neural network interpretation of reference points, which gives a better insight into their nature, along with the necessary modifications to the calculations performed in the forward and backward stages of backpropagation learning.

2. Related work

Distance-based feature extraction (see [12] for an 'umbrella review') is a common practice in many problems concerning non-standard data types. Recently, Iwana et al. [10] proposed a novel approach to extracting distance-based features for time series, where distances between examples and prototypes are calculated using Dynamic Time Warping. The authors propose several candidates for prototypes, e.g., random examples, k-medoids clustering centers, which serve a similar function to the reference points discussed in this paper. Another distance-based feature extraction approach was considered in sleep stage classification using EEG signals, where features were generated based on the distance between each signal and baseline signals [7]. A similar idea was explored in the context of microarray data classification, where the authors use clustering to select prototype genes in order to reduce the number of features [9]. Distance-based features were also explored in text analysis as a way of reducing the dimensionality of data [3] and as a method of similarity learning, where multiple distances between each pair of examples are calculated to produce a set of features used to train a soft classifier and the resulting probability models the similarity between the analyzed objects [13]. Finally, distance-based feature spaces were generated using a typical machine learning apparatus [8], in which a set of classifiers was applied in line with several regular feature selection algorithms to appoint the best sets of features for a collection of datasets. As many as 24 different distance coefficients (not necessarily metrics) were experimented with in this approach. Interestingly, a supervised selection approach, an unsupervised selection approach, as well as an ensemble of them both were presented and examined.

All of the approaches described above use distance-based features, however, all in the context of a particular type of data. Consequently, unlike our research, they do not focus on the nature of such features, but rather on their application to the analyzed problems. Distance-based features in such a general sense have been explored by Tsai et al. [19], where the authors propose two new distance-based features. The features are obtained by first clustering the training dataset into k clusters, where k is the number of classes in the dataset, using the k-means algorithm. Afterwards, the two

features are generated as: 1) the intra-cluster distance between each example and its cluster center and 2) the sum of inter-cluster distances between each example and the remaining cluster centers.

The neural network interpretation presented in this paper reveals a potential relationship between the discussed method and a well established classifier — LVQ [11, 14, 17]. Both involve feature extraction, although LVQ does it in a fully supervised manner and as an intrinsic part of the classification process while the reference points explored in this paper can be created in a semi-supervised or even fully unsupervised fashion and produce features which can be used by any classifier. Nevertheless, the core concept is remarkably similar and it is very interesting to find such a relationship between these two approaches.

3. Feature extraction using distance from reference points

3.1. A general formulation

The idea of creating features based on distances from reference points is not new [3, 7, 10, 19] and in theory is fairly straightforward. It relies on creating k reference points based on the available data and calculating distances between all examples in the dataset and these points. The distance from a given example to a given reference point forms a single feature, giving a total of k features for each example. In the context of classification, the selection of reference points should be performed only on the training data to avoid data leakage. Also, note that this solution does not necessarily require any pre-existing features — a dataset of objects and a distance measure are the only required components.

Formally, given a dataset of arbitrarily defined objects $\mathcal{O} = (o_1, o_2, ..., o_n)$ and a distance measure δ over these objects, the goal is to identify a subset of k reference examples $\mathcal{R} \subseteq \mathcal{O}$ and transform the original objects into a feature space $\vec{X} = (\vec{x_1}, \vec{x_2}, ..., \vec{x_n})$, where each example $\vec{x_i}$ is a vector $\vec{x_i} = (x_{i1}, x_{i2}, ..., x_{ik})$ in which each value x_{ij} encodes the distance from a given object $o_i \in \mathcal{O}$ to a given reference point $r_j \in \mathcal{R}$: $x_{ij} = \delta(o_i, r_j)$.

In this general formulation, any subset of examples fulfills this definition and the objective function needs to be selected for the performed task (e.g., compression, clustering, classification). Since we focus on classification, each dataset has a vector of classes $\vec{y} = (y_1, y_2, ..., y_n)$ and the goal of feature extraction is to maximize the quality of predictions.

3.2. Reference points selection

Before we delve into the specifics of selecting reference points, it is worth addressing a preliminary question: how to determine the optimal number of these points? In certain cases, domain knowledge can guide this decision, allowing us to tailor the

selection closely to the underlying dataset structure. However, domain expertise is often not readily available or sufficiently precise to inform such a choice. A typical approach in this case would be to treat the number of reference points as a hyperparameter and tune it through e.g. cross-validation. Although this approach is perfectly valid, the algorithm we propose in this paper automatically selects the number of reference points k, so there is no need for external inputs or extensive tuning of this hyperparameter. This automatic determination of k not only streamlines the model setup but potentially enhances its robustness by adapting to the dataset's specific characteristics.

There are many possible ways of selecting reference points. The most straightforward is to pick random examples, which we will use as our first baseline to examine whether the reference points selected in other ways have any real impact on the quality of predictions. Our second baseline is a popular approach where the dataset is grouped using some clustering algorithm (e.g., k-means), and the cluster centers serve as reference points [16].

The approach we propose in this paper is based on network centrality and is inspired by research on using this notion for instance weighting in classification [15]. The authors propose a methodology in which they transform the original dataset into a nearest neighborhood graph with distances as edge weights and then use this graph to calculate various network centrality measures of its' vertices. To include the information about class, the authors encode it as the sign of each edge weight, where positive sign signifies that examples belong to the same class, while a negative sign that they belong to different classes. Consequently, centrality measures capable of processing negatively weighted edges need to be used, for which the authors select: weighted degree, status [1], PN [6], and a modified version of clustering coefficient [15]. Since each vertex corresponds to a training example, the calculated centrality scores are later directly used as instance weights during training.

Here, we extend this idea by using the centrality scores to select the best examples for reference points. The whole process is illustrated in Fig. 1 and works as follows. First, the dataset is transformed into a full, weighted graph $G = \langle V, E \rangle$, where each vertex $v_i \in V$ represents a single example $o_i \in O$ and each edge $e_{ij} \in E$ connecting two vertices v_i, v_j has an associated weight $w_{ij} = (2I(y_i = y_j) - 1)\frac{1}{\delta(o_i,o_j)^2}$ reflecting the similarity between the corresponding examples o_i, o_j (expressed as the inverse distance squared with a negative sign if the classes of the examples are different). Given the graph, centrality is calculated for each node. Afterwards, reference points can be picked based on centrality scores of their corresponding nodes and new features can be generated using these points.

Given the above, a standard way to pick the reference examples would simply be to take the top k points according to the chosen measure. However, we argue that, in the discussed context, this common-sense approach is actually deeply flawed and can negatively impact the quality of predictions (as evidenced by our experiments). To better illustrate the issue, consider the example 2D dataset presented in Fig. 2. Consider the top 5 nodes according to degree: 18, 5, 6, 14, 15. Clearly, they are bad candidates for reference points from our method's perspective, because they would produce highly redundant features due to their similar positions. What would be

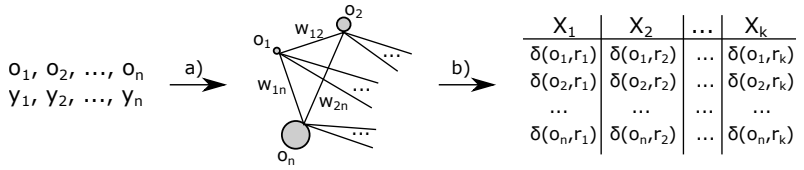


Figure 1. Centrality-based reference points selection. First a) the dataset is transformed into a full weighted graph and centrality of each node is calculated, then b) new features are computed based on k selected reference points $r_1, ..., r_k$.

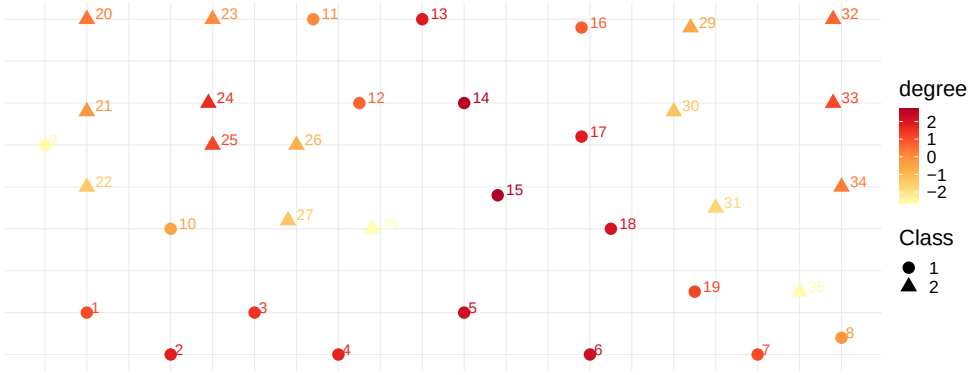
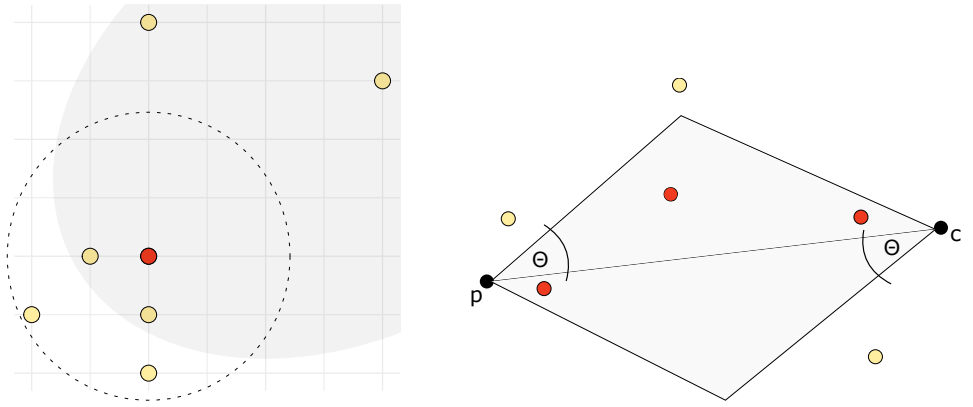


Figure 2. Sample objects with calculated degree. Note that this is only an illustrative example as the process takes place without the knowledge of the underlying space.

preferable in this instance is to look for local maxima (peaks) rather that a global ranking. This would also automatically solve the problem of how many reference points should we select as there are only as many peaks to choose from. In the given example, there are 5 intuitively identifiable peaks: 2, 6, 15, 24, 33, and they are a clearly superior choice for reference points than the top 5 nodes.

To detect peaks in any given function, a natural approach would be to take its derivative. However, going back to our main assumption, we are not dealing with points in space. We only have a distance matrix, which in graph terms translates to a full graph with distances on its edges. A common-sense approach here would be to select peaks based on the nearest neighborhood, i.e., a point is a peak if it is the highest one among its nearest neighbors. However, this approach is also flawed. Consider the example in Fig. 3a. Suppose we want to find 4 nearest neighbors of the red (darker) node. Relying solely on distance, we would get the points circled with the dotted line. However, when deciding whether this point is a peak or not, the more adequate candidates to consider would be the points in the gray area. To be able to find such points, we propose a new, topography-related way of determining the nearest neighborhood, called θ-neighborhood, which we describe in the following subsection.



(a) The problem with classically defined nearest neighborhood

(b) Illustration of the notion of a general direction and a θ-neighborhood

Figure 3. Examples illustrating θ-neighborhood

3.3. θ-neighborhood

The intuition behind the θ-neighborhood is as follows. Given a point p and a different candidate point c, c is the nearest neighbor of p if there is no other point x closer to p in the same "general direction". In other words, we are looking for any point x such that it is closer to p than c and lies in the "general direction" from p to c. If we find it, it means that c is not in the θ-neighborhood of p.

Clearly, the key to this approach is the definition of this "general direction". It cannot be a standard, mathematical definition of direction, since this would only include the points (p, c, and x) lying on the same straight line, which would happen relatively rarely for any given point p, and thus would include nearly all points in its nearest neighborhood. That is why, we formalize the notion of a *general direction* as the surface of an isosceles triangle with the base angle θ as a parameter, as illustrated in Fig. 3b. On a 2D plane (like in the example) the whole area looks like a diamond, in 3D it would look like two cones connected by their base, but in a general case, since we are always considering only 3 nodes at a time — it is an isosceles triangle. This fact is absolutely crucial in our case since we do not actually have any space — we just have distances between objects. In Fig. 3b, observe that with such a definition, c is not in the θ-neighborhood of p because 3 other points (red/darker) are closer to p in the same general direction.

Given the intuitive description, the formal description is rather straightforward. For any point p and a candidate point c, c is **not** in the θ-neighborhood of p if there exists any other point x such that the angle between a line connecting p and c and a line connecting p and x is smaller than θ and the angle between a line connecting p and c and a line connecting c and x is smaller than θ. All remaining points are in the θ-neighborhood of p. Since we already have all the necessary distances, the above is easy to calculate using the Law of cosines. Formally, c is in the θ-neighborhood of p

if there does not exist any other point x satisfying all of the following conditions:

$$\delta(p,x) < \delta(p,c) \tag{1}$$

$$\delta(c,x) < \delta(p,c) \tag{2}$$

$$\frac{\delta(p,c)^2 + \delta(p,x)^2 - \delta(c,x)^2}{2 \cdot \delta(p,c) \cdot \delta(p,x)} > \cos\theta \tag{3}$$

$$\frac{\delta(p,c)^2 + \delta(c,x)^2 - \delta(p,x)^2}{2 \cdot \delta(p,c) \cdot \delta(c,x)} > \cos\theta \tag{4}$$

3.4. Computational complexity

The computational complexity of our method can be broken down into several key components: the transformation of the dataset into a full weighted graph, the calculation of centrality measures, the selection of reference points based on these measures, and the generation of new features using the selected reference points.

- **Graph construction** The initial step involves transforming the dataset into a full weighted graph $G = \langle V, E \rangle$. Given n examples in the dataset, this step requires calculating the pairwise distance between all examples, resulting in a complexity of $O(n^2)$ for the distance calculations. The construction of the graph itself, including setting up vertices and edges with associated weights, also operates within $O(n^2)$ complexity, as each of the $n(n-1)/2$ possible edges between vertices is considered.

- **Centrality Calculation** The complexity of calculating centrality measures depends on the specific algorithm used. For the purpose of this analysis, we consider algorithms that operate in $O(n^2)$ for a full graph, acknowledging that specific implementations may vary in complexity.

- **Reference Points Selection** The selection of reference points based on centrality scores involves either sorting the nodes by their scores or applying other criteria like the proposed θ-neighborhood to ensure diversity among the selected points. The sorting operation has a complexity of $O(n \log n)$, but the proposed θ-neighborhood approach involves checking all possible triplets of points, so has a complexity of $O(n^3)$.

- **Feature Generation** For each example in the dataset, distances to the k selected reference points are calculated to form new features. This step has a complexity of $O(nk)$, which is linear with respect to the number of examples and the number of reference points.

The overall computational complexity of our method is dominated by the reference points selection, which in our proposal is $O(n^3)$. This complexity indicates that while the method is computationally feasible for small to medium-sized datasets, its scalability to larger datasets may be constrained by available computational resources. However, it is important to note that our method is particularly designed to

Table 1. Dataset characteristics. The last column shows the number of instances in the majority class, as well as its share in the dataset.

Dataset	Instances	Features	Classes	Majority class
breast-cancer-wisconsin	699	10	2	458 (65.5%)
ecoli	336	8	8	143 (42.6%)
glass	214	10	6	76 (35.5%)
image-segmentation	2310	20	7	30 (14.3%)
ionosphere	351	35	2	225 (64.1%)
iris	150	5	3	50 (33.3%)
optdigits	5620	65	10	572 (10.2%)
pima-indians-diabetes	768	9	2	500 (65.1%)
sonar.all	208	61	2	111 (53.4%)
spectrometer	531	102	48	55 (10.4%)
statlog-satimage	6435	37	6	1533 (23.8%)
statlog-vehicle	846	19	4	217 (25.7%)
vowel-context	990	11	11	90 (09.1%)
wine	178	14	3	71 (39.9%)
yeast	1484	9	10	463 (31.2%)

enhance feature extraction in scenarios where traditional methods may struggle, such as with datasets that lack preexisting features or where domain knowledge is limited. For very large datasets, optimizations such as approximate nearest neighbor search, graph sparsification, or parallel processing techniques could be employed to mitigate computational demands.

4. Experimental evaluation

We have 3 main goals in our experimental evaluation: a) to compare the performance of peak- and top-scoring centrality-based reference points; b) to compare the performance of various centrality measures; c) to comparatively evaluate centrality-based, clustering-based, and random reference points.

4.1. Setup

In the experiments, we use 15 publicly available datasets [5] with various characteristics, described collectively in Table 1. The Table shows, in particular, the 'majority class', which is the percentage of objects from the most frequent class in the dataset. This percentage serves as a reference for the classification accuracy, which – to be recognized as valid – should exceed the 'majority class'. Notice that the characteristics of the used datasets are quite diverse in terms of the number of instances, features, classes, as well as in terms of data balancing.

The experimental procedure for each approach on each dataset is as follows. First, the dataset is split into 70% training and 30% testing parts. Then, reference points are extracted from training data and new features are generated in training and testing sets. Notice that after this step only these new features are used in all the analyses (the original features were simply used to calculate the distances). Since the peak-based

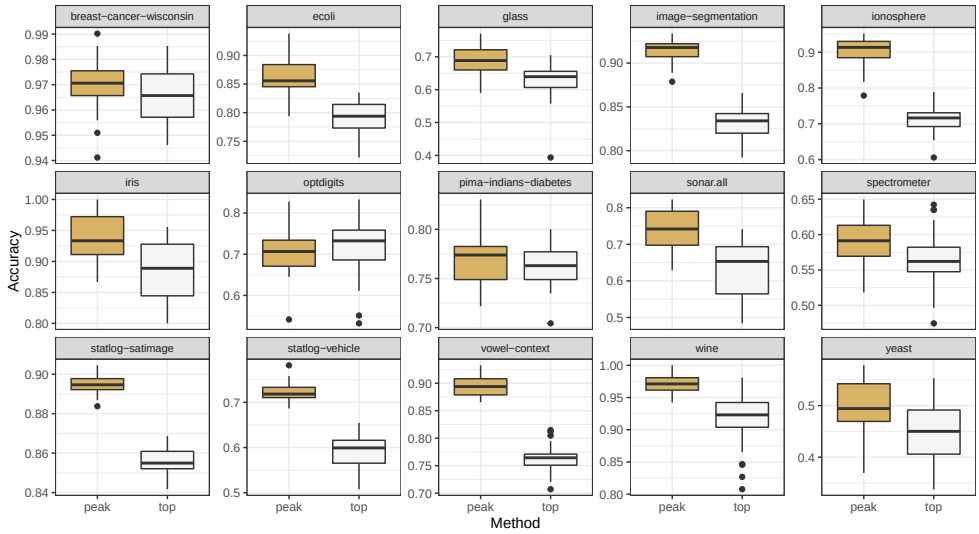


Figure 4. Accuracy of reference points selected using peak and top-k degree. Each boxplot presents the median, 1st and 3rd quartile along with the interquartile region (IQR), the whiskers extend 1.5*IQR above and below the quartiles, and the points falling outside that range are outliers.

approach selects the number of points k automatically, we run it first and then use the determined value of k in all other approaches for a fair comparison. Afterwards, we tune the hyperparameters of the classifier by performing 2-fold cross-validation repeated 5 times over the training data. Then, the classifier is finally trained on new features on the training data with selected hyperparameters and tested on the hold-out testing part. The whole process is repeated 30 times to report a more reliable testing score. The evaluation metric used in all experiments is standard accuracy. If not stated otherwise, in all experiments we use linear SVM and Euclidean distance. The code and results are publicly available for reproducibility[1].

4.2. Peak degree vs top-k degree

The aim of the first experiment is to validate our claim about the superiority of using peak-scoring- rather than top-scoring-examples for reference points. Since the peaks approach determines the number of reference points automatically, in each experiment we first run the peaks approach and pass the number of peaks to the top-k approach. We used weighted degree as the centrality measure.

The results presented in Fig. 4 illustrate that the reference points selected using peak degree are clearly superior to those selected using top-k degree. The statistical

[1] `https://github.com/MaciejPiernik/featurescope`

Table 2. Number of reference points selected by each centrality measure using the procedure of peak detection.

Dataset	k_{degree}	k_{PN}	k_{status}
breast-cancer-wisconsin	10	52	17
ecoli	11	36	11
glass	15	22	14
image-segmentation	30	50	27
ionosphere	8	28	7
iris	7	17	8
optdigits	7	186	66
pima-indians-diabetes	20	71	27
sonar.all	6	13	5
spectrometer	48	48	20
statlog-satimage	90	470	245
statlog-vehicle	30	67	32
vowel-context	50	52	16
wine	8	16	12
yeast	6	34	56

analysis using a t-test for each dataset separately reveals that the peaks approach was significantly better than top-k in 12 cases while the reverse was never true ($\alpha = 0.05$). Moreover, the Wilcoxon signed ranks test confirms that, overall, the peaks approach is significantly better than top-k ($p < 0.001$). This is a very strong result which showcases the power of the θ-neighborhood and our peak selection method.

This result highlights the problem with the top-k approach we discussed in Section 3.2, namely, that it tends to select points very close to each other, particularly with measures such as the degree, what leads to redundant features. We introduced the θ-neighborhood and the peaks approach precisely to address this issue and this result clearly confirms our line of reasoning. Since the outcome of this experiment is so conclusive, we will rely on the peaks approach in all of the remaining experiments.

4.3. Comparative evaluation of centrality measures

The aim of the second experiment is to see what impact does the selected centrality measure have on the quality of predictions. As already described in Section 3.2, we are considering 4 measures: weighted degree, status [1], PN [6], and a modified version of clustering coefficient [15], though only the first three are reported. This is because the results of clustering coefficient are identical to the results achieved by the weighted degree, which may be explained by the fact that in a full graph these coefficients (despite producing different values) are ordinarily equivalent. The results of the comparison are presented in Table 2 and Fig. 5.

The results show that, for most datasets, the selected centrality measure does not impact the quality of predictions in a consistent way. Nevertheless, there are a few datasets in which a significant difference can be observed. A pairwise t-test ($\alpha = 0.05$) reveals that PN significantly outperforms degree on 5 datasets (image-segmentation, statlog-vehicle, statlog-satimage, yeast, optdigits) as well as status on 4 datasets (image-segmentation, ionosphere, statlog-vehicle, vowel-context). Addi-

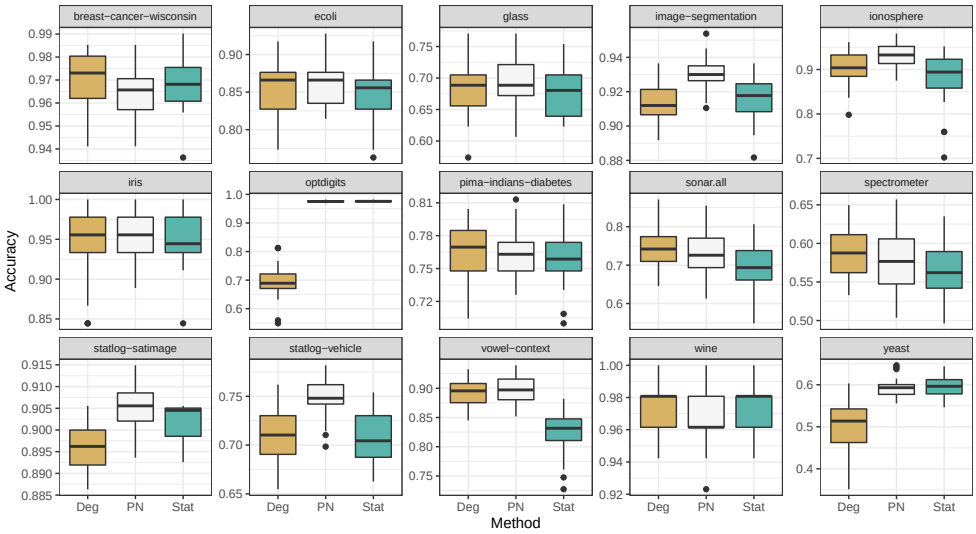


Figure 5. Accuracy of reference points selected using: degree, PN, and Status. Each boxplot presents the median, 1st and 3rd quartile along with the interquartile region (IQR), the whiskers extend 1.5*IQR above and below the quartiles, and the points falling outside that range are outliers.

tionally, status significantly outperforms degree on 2 datasets (yeast and optdigits) while the reverse is true for 1 dataset (sonar.all). However, analyzing the results collectively, the Friedman statistical test did not reveal any significant differences between the approaches. Moreover, the above-mentioned datasets do not seem to share any common part understood as similar number of classes, features or data balancing, suggesting no relation between dataset characteristics and centrality measures' impact on the quality of predictions.

In addition to the statistical analysis, one cannot draw conclusions based only on these accuracies because the number of reference points selected by each method is different. In fact, when looking at the number of points selected by each approach (see Table 2), all of these differences can be attributed to this factor, although in some cases, even though the difference in the number of points is very large (e.g., breast-cancer-wisconsin), it does not impact the quality of predictions. A probable explanation for this outcome is that the number of points necessary for achieving high quality predictions is dependent on the dataset and, after a certain threshold, adding more features does not improve the quality further. Regardless of this observation, the results show that there is no statistically significant difference between the analyzed approaches.

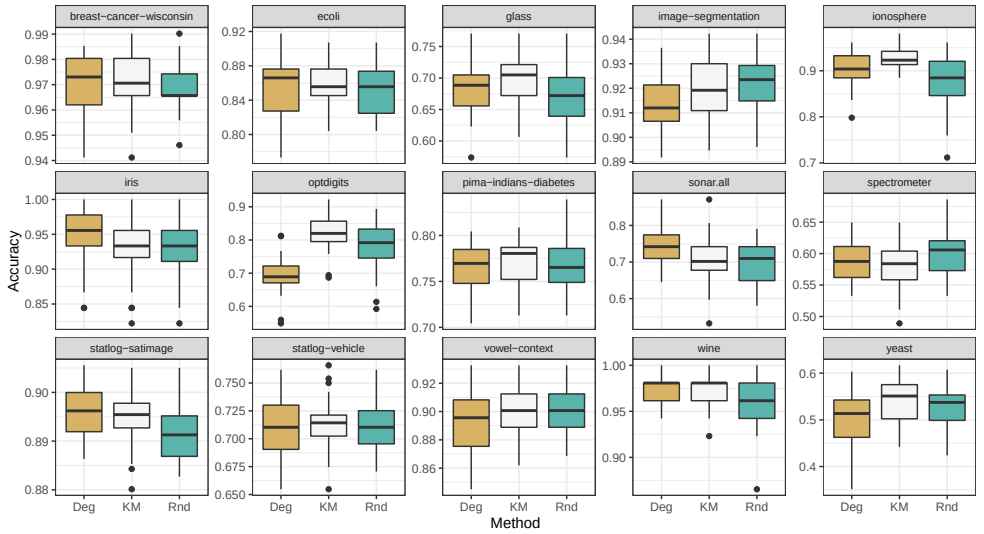


Figure 6. Accuracy of reference points selected using: centrality, clustering, and randomly. Each boxplot presents the median, 1st and 3rd quartile along with the interquartile region (IQR), the whiskers extend 1.5*IQR above and below the quartiles, and the points falling outside that range are outliers.

4.4. Comparative evaluation of reference points selection methods

The goal of our final experiment is to compare 3 methods for selecting reference points: network centrality, clustering, and random. We will rely on peak degree for the centrality method while for clustering we will use the k-medoids algorithm as it naturally picks the cluster centers from the provided examples. To make sure each approach is treated fairly, in each of the 30 runs we run the centrality-based method first and then use the selected number of points in the remaining two approaches. The results of this comparison are presented in Fig. 6.

The results of this experiment are very surprising as all of the compared approaches perform very similarly. Analyzing the results separately using a pairwise t-test, some significant differences can be identified, however, they are very inconsistent and do not bring much insight into the analysis. The clustering approach significantly outperformed random points on 3 datasets (glass, ionosphere, optdigits) and degree on 4 datasets (image-segmentation, ionosphere, yeast, optdigits). Degree significantly outperformed clustering on 1 dataset (sonar.all) and random points on 3 datasets (ionosphere, sonar.all, wine). Random points significantly outperformed degree on 3 datasets (image-segmetation, yeast, optdigits). Analyzing the results collectively, the Friedman test could not significantly distinguish between the analyzed approaches.

In summary, the results show that there is no difference in the predictive performance between the compared approaches and, together with the results of the two previous experiments, they lead us to a very important conclusion. The number of

reference points is far more important than their exact positions, given that they are reasonably spread across the data.

5. Neural network interpretation of reference points — the Euclidean neuron

The reference points discussed in this paper have an interesting interpretation which opens doors to optimization-based reference points selection. This interpretation works in a constrained case where each object is a real-valued vector with m features. From a neural network perspective, the role of the discussed method is to find an embedding of the input space $\mathcal{R}_m$ into $\mathcal{R}_k$, where each new dimension corresponds to the Euclidean distance from one of the k selected points. So, for every input vector $\vec{x}$ and a selected point $\vec{w}$, the respective dimension is generated as $||\vec{x} - \vec{w}||$. This can be viewed as a layer of a neural network in which, for each single neuron, $\vec{x}$ are the inputs and $\vec{w}$ the weights, however, without the typical dot product and the activation function. In our case, there is no need for the activation function since nonlinearity is introduced by the Euclidean distance. The dot product and the activation function are together substituted with $||\vec{x} - \vec{w}||$. The only difference between this layer and a typical one is that it aggregates the results a little differently.

The network resulting from the above-described interpretation can be trained with only a slightly modified backpropagation algorithm. In the feed-forward phase, when calculating the output of a neuron, instead of $\phi(\vec{x} \bullet \vec{w})$ we have $||\vec{x} - \vec{w}||$, where $\bullet$ denotes the dot product. In the backward phase, originally, for each weight w_i we are calculating its impact on the total error as $\frac{\partial Error}{\partial w_i}$. This is done by expanding this expression using the chain rule: $\frac{\partial Error}{\partial w_i} = \frac{\partial Error}{\partial out}\frac{\partial out}{\partial sig}\frac{\partial sig}{\partial w_i}$, where sig (signal) is the linear combination of inputs to a given neuron ($sig = \sum_{i=0}^{m} w_i x_i$) and out is the signal passed through some activation function (e.g., $\sigma(sig)$). Since we do not have separate signal and activation, we get: $\frac{\partial Error}{\partial w_i} = \frac{\partial Error}{\partial dist}\frac{\partial dist}{\partial w_i}$, where $dist = \sqrt{\sum_{i=1}^{m}(x_i - w_i)^2}$. The first derivative does not change at all from the original setting, since in our case $dist$ is simply the output of the layer. As for the second layer, treating $dist$ as another composite function $dist = \sqrt{tmp}$ and applying the chain rule again we obtain: $\frac{\partial dist}{\partial w_i} = \frac{\partial \sqrt{tmp}}{\partial tmp}\frac{\partial tmp}{\partial w_i} = \frac{1}{2\sqrt{tmp}}2(x_i - w_i) = \frac{x_i - w_i}{\sqrt{tmp}} = \frac{x_i - w_i}{dist}$. Applying these two modifications (one in the feed-forward and one in the backward phase) results in a fully trainable neuron which can be composed into layers, just like regular neurons (our implementation as a Keras [4] layer is publicly available[2]). We call it the *Euclidean neuron*.

When drawing a parallel between distance-based approaches and neural networks, it is also worth talking about similarity-based methods, particularly those employing Radial Basis Function (RBF) kernels [2]. RBF kernels are a popular choice in various machine learning algorithms, including support vector machines and certain neural networks, due to their ability to transform the feature space in a way that makes

[2] `https://github.com/MaciejPiernik/featurescope`

non-linearly separable data more separable.

The core difference between the distance-based feature extraction discussed in this paper and the RBF kernel lies in the treatment of distances. In our approach, the distance itself is used directly as a feature, while the RBF kernel transforms the distance into a measure of similarity. This transformation introduces a non-linear component to the model, enabling it to capture complex patterns in the data. Despite this difference, both approaches share a common ground in their reliance on the concept of distance or proximity between data points. While our method uses distance directly to construct a new feature space, RBF kernels use a function of distance to map the data into a higher-dimensional space where it becomes more amenable to linear classification methods. This highlights an interesting intersection between distance-based and similarity-based methods, suggesting potential avenues for integrating these concepts to leverage the strengths of both approaches in future research.

Unlike the reference points considered in this paper, the neural network interpretation leads to a very different family of reference points, namely, based on direct optimization. Although such an approach is a departure from the main theme of this paper (we do not rely on the input space, only on distances), we think it is important to mention it here because it presents an interesting direction for future research, particularly in the light of our experimental results.

6. Conclusions and future work

In this paper, we studied the concept of feature extraction using distance from reference points. We formalized the framework for reference-points-based feature extraction and proposed its instantiation based on network centrality. To achieve this goal, we proposed a novel way of defining the nearest neighborhood, called θ-neighborhood, which allows for navigating the topography of fully connected graphs. We also presented a neural network interpretation of reference points together with the notion of the Euclidean neuron, which paves the way for optimization-based reference points selection.

The results of the experiments show that the developed method for identifying peaks produces significantly better reference points than a traditional top-k approach. Moreover, the experiments reveal that the number of reference points seems to have a much bigger impact on the quality of the resulting features (from the classification perspective) than the selection method, as the tested variants (4 centrality-based and 1 clustering) showed no significant improvement over random points.

Due to a wide range of covered topics, this research opens many new paths of further research. The main such path is a further exploration of distance-based topographies, as the θ-neighborhood introduced in this paper is only one of many possible ways of addressing this problem. The second path is the neural network interpretation of reference points and the Euclidean neuron, as it needs an experimental evaluation and could possibly form a new family of networks. Another important avenue is employing various heuristics to reduce the high computational demands of the pro-

posed approach. Finally, since we compared random, clustering- and centrality-based reference points, other selection methods could be explored.

Acknowledgment

This research has been partially supported by the statutory funds of Poznan University of Technology.

References

[1] Bonacich P. and Lloyd P. Calculating status with negative relations. *Social Networks - SOC NETWORKS*, 26:331–338, 2004.

[2] Buhmann M. D. *Radial Basis Functions: Theory and Implementations*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[3] Canuto S., Sousa D., Goncalves M., and Rosa T. A thorough evaluation of distance-based meta-features for automated text classification. *IEEE Transactions on Knowledge & Data Engineering*, 30(12), 2018.

[4] Chollet F. et al. Keras. `https://keras.io`, 2015.

[5] Dua D. and Graff C. Uci machine learning repository. `http://archive.ics.uci.edu/ml`, 2017.

[6] Everett M. and Borgatti S. Networks containing negative ties. *Social Networks*, 38:111–120, 2014.

[7] Gharbali A., Najdi S., and Fonseca J. Investigating the contribution of distance-based features to automatic sleep stage classification. *Computers in Biology and Medicine*, 96, 2018.

[8] Hallajian B., Motameni H., and Akbari E. Ensemble feature selection using distance-based supervised and unsupervised methods in binary classification. *Expert Systems with Applications*, 200:116794, 2022.

[9] Hanczar B., Courtine M., Benis A., Hennegar C., Clément K., and Zucker J.-D. Improving classification of microarray data using prototype-based feature selection. *SIGKDD Explor. Newsl.*, 5(2):23–30, 2003.

[10] Iwana B. and Uchida S. Time series classification using local distance-based features in multi-modal fusion networks. *Pattern Recognition*, 97:107024, 2019.

[11] Kohonen T. *Learning Vector Quantization*, pages 175–189. Springer, 1995.

[12] Linja J., Hämäläinen J., Nieminen P., and Kärkkäinen T. Feature selection for distance-based regression: An umbrella review and a one-shot wrapper. *Neurocomputing*, 518:344–359, 2023.

[13] López-Iñesta E., Grimaldo F., and Arevalillo-Herráez M. Comparing feature-based and distance-based representations for classification similarity learning. volume 269 of *Frontiers in Artificial Intelligence and Applications*, 2014.

[14] Nova D. and Estévez P. A. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3):511–524, 2014.

[15] Piernik M., Brzezinski D., Morzy T., and Morzy M. Using network analysis to improve nearest neighbor classification of non-network data. In *Proceedings of the 23rd International Symposium on Foundations of Intelligent Systems*, volume 10352 of *LNCS*, pages 105–115, 2017.

[16] Piernik M. and Morzy T. A study on using data clustering for feature extraction to improve the quality of classification. *Knowledge and Information Systems*, 63:1771–1805, 2021.

[17] Ravichandran J., Kaden M., and Villmann T. Variants of recurrent learning vector quantization. *Neurocomputing*, 502:27–36, 2022.

[18] Sathe S. and Aggarwal C. C. Similarity forests. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 395–403. ACM, 2017.

[19] Tsai C.-F., Lin W.-Y., Hong Z.-F., and Hsieh C.-Y. Distance-based features in pattern classification. *EURASIP Journal on Advances in Signal Processing*, 2011, 2011.

Received 08.11.2023, Accepted 10.06.2024