Explaining the Predictions of Unsupervised Learning Models

Grégoire Montavon[1,2][0000−0001−7243−6186], Jacob Kauffmann[1][0000−0003−2667−513X], Wojciech Samek[3,2][0000−0002−6283−3265], and Klaus-Robert Müller[1,2,4,5][0000−0002−3861−7685]

[1] ML Group, Department of Electrical Engineering and Computer Science, Technische Universität Berlin, Berlin, Germany
[2] BIFOLD – Berlin Institute for Foundations of Learning and Data, Berlin, Germany
[3] Department of Artificial Intelligence, Fraunhofer Heinrich Hertz Institute, Berlin, Germany
[4] Department of Artificial Intelligence, Korea University, Seoul, Korea
[5] Max Planck Institut für Informatik, Saarbrücken, Germany

Abstract. Unsupervised learning is a subfield of machine learning that focuses on learning the structure of data without making use of labels. This implies a different set of learning algorithms than those used for supervised learning, and consequently, also prevents a direct transposition of Explainable AI (XAI) methods from the supervised to the less studied unsupervised setting. In this chapter, we review our recently proposed 'neuralization-propagation' (NEON) approach for bringing XAI to workhorses of unsupervised learning such as kernel density estimation and k-means clustering. NEON first converts (without retraining) the unsupervised model into a functionally equivalent neural network so that, in a second step, supervised XAI techniques such as layer-wise relevance propagation (LRP) can be used. The approach is showcased on two application examples: (1) analysis of spending behavior in wholesale customer data and (2) analysis of visual features in industrial and scene images.

Keywords: Explainable AI · Unsupervised Learning · Neural Networks.

1 Introduction

Supervised learning has been in the spotlight of machine learning research and applications for the last decade, with deep neural networks achieving record-breaking classification accuracy and enabling new machine learning applications [23, 5, 15]. The success of deep neural networks can be attributed to their ability to implement with their multiple layers, complex nonlinear functions in a compact manner [32]. Recently, a significant amount of work has been dedicated to make deep neural network models more transparent [24, 40, 13, 41], for example, by proposing algorithms that identify which input features are responsible for a given classification outcome. Methods such as layer-wise relevance propagation

(LRP) [3], guided backprop [47], and Grad-CAM [42], have been shown capable of quickly and robustly computing these explanations.

Unsupervised learning is substantially different from supervised learning in that there is no ground-truth supervised signal to match. Consequently, non-neural network models such as kernel density estimation or k-means clustering, where the user controls the scale and the level of abstraction through a particular choice of kernel or feature representation, have remained highly popular. Despite the predominance of unsupervised machine learning in a variety of applications (e.g. [22, 9]), research on explaining unsupervised models has remained relatively sparse [25, 30, 28, 19, 18] compared to their supervised counterparts. Paradoxically, it might in fact be unsupervised models that most strongly require interpretability. Unsupervised models are indeed notoriously hard to quantitatively validate [51], and the main purpose of applying these models is often to better understand the data in the first place [17, 9].

In this chapter, we review the 'neuralization-propagation' (NEON) approach we have developed in the papers [19, 18, 20] to make the predictions of unsupervised models, e.g. cluster membership or anomaly score, explainable. NEON proceeds in two steps: (1) the decision function of the unsupervised model is reformulated (without retraining) as a functionally equivalent neural network (i.e. it is 'neuralized'); (2) the extracted neural network structure is then leveraged by the LRP method to produce an explanation of the model prediction. We review the application of NEON to kernel density estimation for outlier detection and k-means clustering, as presented originally in [19, 18, 20]. We also extend the reviewed work with a new contribution: explanation of *inlier* detection, and we use the framework of random features [36] for that purpose.

The NEON approach is showcased on several practical examples, in particular, the analysis of wholesale customer data, image-based industrial inspection, and analysis of scene images. The first scenario covers the application of the method directly to the raw input features, whereas the second scenario illustrates how the framework can be applied to unsupervised models built on some intermediate layer of representation of a neural network.

2 A Brief Review of Explainable AI

The field of Explainable AI (XAI) has produced a wealth of explanation techniques and types of explanation. They address the heterogeneity of ML models found in applications and the heterogeneity of questions the user may formulate about the model and its predictions. An explanation may take the form of a simple decision tree (or other intrinsically interpretable model) that approximates the model's input-output relation [10, 29]. Alternatively, an explanation may be a prototype for the concept represented at the output of the model, specifically, an input example to which the model reacts most strongly [45, 34]. Lastly, an explanation may highlight what input features are the most important for the model's predictions [7, 4, 3].

In the following, we focus on a well-studied problem of XAI, which is how to attribute the prediction of an individual data point, to the input features [4, 48, 45, 3, 37, 29, 50]. Let us denote by $\mathcal{X} = \mathcal{I}_1 \times \cdots \times \mathcal{I}_d$ the input space formed by the concatenation of d input features (e.g. words, pixels, or sensor measurements). We assume a learned model $f : \mathcal{X} \to \mathbb{R}$ (supervised or unsupervised), mapping each data point in $\mathcal{X}$ to a real-valued score measuring the evidence for a class or some other predicted quantity. The problem of attribution can be abstracted as producing for the given function f a mapping $\mathcal{E}_f : \mathcal{X} \to \mathbb{R}^d$ that associates to each input example a vector of scores representing the (positive or negative) contribution of each feature. Often, one requires attribution techniques to implement a *conservation* (or completeness) property, where for all $\boldsymbol{x} \in \mathcal{X}$ we have $\mathbf{1}^\top \mathcal{E}_f(\boldsymbol{x}) = f(\boldsymbol{x})$ i.e. for every data point the sum of explanation scores over the input features should match the function value.

2.1 Approaches to Attribution

A first approach, *occlusion-based*, consists of testing the function to explain against various occlusions of the input features [53, 54]. An important method of this family (and which was originally developed in the context of game theory) is the Shapley value [43, 48, 29]. The Shapley value identifies a unique attribution that satisfies some predefined set of axioms of an explanation, including the conservation property stated above. While the approach has strong theoretical underpinnings, computing the explanation however requires an exponential number of function evaluations (an evaluation for every subset of input features). This makes the Shapley value in its basic form intractable for any problem with more than a few input dimensions.

Another approach, *gradient-based*, leverages the gradient of the function, so that a mapping of the function value onto the multiple input dimensions is readily obtained [45, 50]. The method of integrated gradients [50], in particular, attributes the prediction to input features by integrating the gradient along a path connecting some reference point (e.g. the origin) to the data point. The method requires somewhere between ten and a hundred function evaluations, and satisfies the aforementioned conservation property. The main advantage of gradient-based methods is that, by leveraging the gradient information in addition to the function value, one no longer has to perturb each input feature individually to produce an explanation.

A further approach, *surrogate-based*, consists of learning a simple local surrogate model of the function which is as accurate as possible, and whose structure makes explanation fast and unambiguous [37, 29]. For example, when approximating the function locally with a linear model, e.g. $g(\boldsymbol{x}) = \sum_{i=1}^{d} x_i w_i$, the output of that linear model can be easily decomposed to the input features by taking the individual summands. While explanation itself is fast to compute, training the surrogate model incurs a significant additional cost, and further care must be taken to ensure that the surrogate model implements the same

decision strategy as the original model, in particular, that it uses the same input features.

A last approach, *propagation-based*, assumes that the prediction has been produced by a neural network, and leverages the neural network structure by casting the problem of explanation as performing a backward pass in the network [47, 3, 42]. The propagation approach is embodied by the Layer-wise Relevance Propagation (LRP) method [3, 31]. The backward pass implemented by LRP consists of a sequence of conservative propagation steps where each step is implemented by a propagation rule. Let j and k be indices for neurons at layer l and $l+1$ respectively, and assume that the function output $f(\boldsymbol{x})$ has been propagated from the top-layer to layer $l+1$. We denote the resulting attribution onto these neurons as the vector of 'relevance scores' $(R_k)_k$. LRP then defines 'messages' $R_{j\leftarrow k}$ that redistribute the relevance R_k to neurons in the layer below. These messages typically have the structure $R_{j\leftarrow k} = [z_{jk}/\sum_j z_{jk}] \cdot R_k$, where z_{jk} models the contribution of neuron j to activating neuron k. The overall relevance of neuron j is then obtained by computing $R_j = \sum_k R_{j\leftarrow k}$. It is easy to show that application of LRP from one layer to the layer below is conservative. Consequently, the explanation formed by iterating the LRP propagation from the top layer to the input layer is therefore also conservative, i.e. $\sum_i R_i = \cdots = \sum_j R_j = \sum_k R_k = \cdots = f(\boldsymbol{x})$. As a result, explanations satisfying the conservation property can be obtained within a single forward/backward pass, instead of multiple function evaluations, as it was the case for the approaches described above. The runtime advantage of LRP facilitates explanation of large models and datasets (e.g. GPU implementations of LRP can achieve hundreds of image classification explanations per second [1, 40]).

2.2 Neuralization-Propagation

Propagation-based explanation techniques such as LRP have a computational advantage over approaches based on multiple function evaluations. However, they assume a preexisting neural network structure associated to the prediction function. Unsupervised learning models such as kernel density estimation or k-means, are a priori not neural networks. However, the fact that these models are not given as neural networks does not preclude the existence of a neural network that implements the same function. If such a network exists (neural network equivalents of some unsupervised models will be presented in Sections 3 and 4), we can quickly and robustly compute explanations by applying the following two steps:

Step 1: The unsupervised model is '*neuralized*', that is, rewritten (without retraining) as a functionally equivalent neural network.

Step 2: The LRP method is applied to the resulting neural network, in order to produce an explanation of the prediction of the original model.

These two steps are illustrated in Fig. 1. In practice, for the second step to work well, some restrictions must be imposed on the type of neurons composing the network. In particular neurons should have a clear directionality in

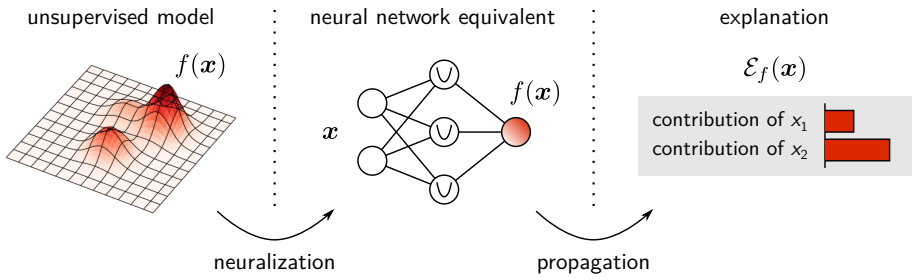

Fig. 1. Overview of the neuralization-propagation (NEON) approach to explain the predictions of an unsupervised model. As a first step, the unsupervised model is transformed without retraining into a functionally equivalent neural network. As a second step, the LRP procedure is applied to identify, with help of the neural network structure, by what amount each input feature has contributed to a given prediction.

their input space to ensure that meaningful propagation to the lower layer can be achieved. (We will see in Sections 3 and 4, that this requirement does not always hold.) Hence, the 'neuralized model' must be designed under the double constraint of (1) replicating the decision function of the unsupervised model exactly, and (2) being composed of neurons that enable a meaningful redistribution from the output to the input features.

3 Kernel Density Estimation

Kernel density estimation (KDE) [35] is one of the most common methods for unsupervised learning. The KDE model (or variations of it) has been used, in particular, for anomaly detection [26, 21, 38]. It assumes an unlabeled dataset $\mathcal{D} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N)$, and a kernel, typically the Gaussian kernel $\mathbb{K}(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$. The KDE model predicts a new data point $\boldsymbol{x}$ by computing:

$$\tilde{p}(\boldsymbol{x}) = \frac{1}{N} \sum_{k=1}^{N} \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{u}_k\|^2). \tag{1}$$

The function $\tilde{p}(\boldsymbol{x})$ can be interpreted as an (unnormalized) probability density function. From this score, one can predict inlierness or outlierness of a data point. For example, one can say that $\boldsymbol{x}$ is more anomalous than $\boldsymbol{x}'$ if the inequality $\tilde{p}(\boldsymbol{x}) < \tilde{p}(\boldsymbol{x}')$ holds. In the following, we consider the task of neuralizing the KDE model so that its inlier/outlier predictions can be explained.

3.1 Explaining Outlierness

A first question to ask is why a particular example $\boldsymbol{x}$ is predicted by KDE to be an *outlier*, more specifically, what features of this example contribute to outlierness. As a first step, we consider what is a suitable measure of outlierness. The function

$\tilde{p}(\boldsymbol{x})$ produced by KDE decreases with outlierness, and also saturates to zero even though outlierness continues to grow. A better measure of outlierness is given by [19]:

$$o(\boldsymbol{x}) \triangleq -\frac{1}{\gamma} \log \tilde{p}(\boldsymbol{x}),$$

Unlike the function $\tilde{p}(\boldsymbol{x})$, the function $o(\boldsymbol{x})$ increases as the probability decreases. It also does not saturate as $\boldsymbol{x}$ becomes more distant from the dataset. We now focus on neuralizing the outlier score $o(\boldsymbol{x})$. We find that $o(\boldsymbol{x})$ can be expressed as the two-layer neural network:

$$\begin{aligned} h_k &= \|\boldsymbol{x} - \boldsymbol{u}_k\|^2 && \text{(layer 1)} \\ o(\boldsymbol{x}) &= \text{LME}_k^{-\gamma}\{h_k\} && \text{(layer 2)} \end{aligned}$$

where $\text{LME}_k^{\alpha}\{h_k\} = \frac{1}{\alpha} \log\left(\frac{1}{N}\sum_{k=1}^{N} \exp(\alpha\, h_k)\right)$ is a generalized log-mean-exp pooling. The first layer computes the square distance of the new example from each point in the dataset. The second layer can be interpreted as a soft min-pooling. The structure of the outlier computation is shown for a one-dimensional toy example in Fig. 2.

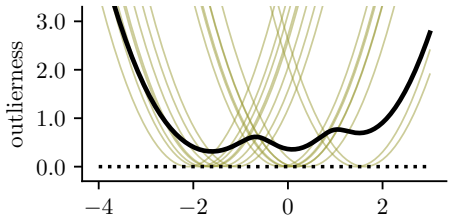

Fig. 2. Neuralized view of kernel density estimation for outlier prediction. The outlier function can be represented as a soft min-pooling over square distances. These distances also provide directionality in input space.

This structure is particularly amenable to explanation. In particular, redistribution of $o(\boldsymbol{x})$ in the intermediate layer can be achieved by a soft argmin operation, e.g.

$$R_k = \frac{\exp(-\beta h_k)}{\sum_k \exp(-\beta h_k)} \cdot o(\boldsymbol{x}),$$

where β is a hyperparameter to be selected. Then, propagation on the input features can leverage the geometry of the distance function, by computing

$$R_i = \sum_k \frac{[\boldsymbol{x} - \boldsymbol{u}_k]_i^2}{\epsilon + \|\boldsymbol{x} - \boldsymbol{u}_k\|^2} R_k.$$

The hyperparameter ϵ in the denominator is a stabilization term that 'dissipates' some of the relevance when $\boldsymbol{x}$ and $\boldsymbol{u}_k$ coincide.

Referring back to Section 2.1 we want to stress that computing the relevance of input features with LRP has the same computational complexity as a single forward pass, and does not require to train an explainable surrogate model.

3.2 Explaining Inlierness: Direct Approach

In Section 3.1, we have focused on explaining what makes a given example an outlier. An equally important question to ask is why a given example $\boldsymbol{x}$ is predicted by the KDE model to be an *inlier*. Inlierness is naturally modeled by the KDE output $\tilde{p}(\boldsymbol{x})$. Hence we can define the measure of inlierness as $\mathtt{i}(\boldsymbol{x}) \triangleq \tilde{p}(\boldsymbol{x})$. An inspection of Eq. (1) suggests the following two-layer neural network:

$$\begin{aligned} h_k &= \exp(-\gamma \left\|\boldsymbol{x} - \boldsymbol{u}_k\right\|^2) && \text{(layer 1)} \\ \mathtt{i}(\boldsymbol{x}) &= \tfrac{1}{N} \textstyle\sum_{k=1}^{N} h_k && \text{(layer 2)} \end{aligned}$$

The first layer performs a mapping on Gaussian functions at different locations, and the second layer performs an average pooling. We now consider the task of propagation. A natural way of redistributing in the top layer is in proportion to the activations. This gives us the scores

$$R_k = \frac{h_k}{\sum_k h_k} \mathtt{i}(\boldsymbol{x}).$$

A decomposition of R_k on the input features is however difficult. Because the relevance R_k can be rewritten as a product:

$$R_k = \frac{1}{N} \prod_{i=1}^{d} \exp(-\gamma \, (x_i - u_{ik})^2)$$

and observing that the contribution R_k can be made nearly zero by perturbing any of the input features significantly, we can conclude that every input feature contributes equally to R_k and should therefore be attributed an equal share of it. Application of this strategy for every neuron k would result in an uniform redistribution of the score $\mathtt{i}(\boldsymbol{x})$ to the input features. The explanation would therefore be qualitatively always the same, regardless of the data point $\boldsymbol{x}$ and the overall shape of the inlier function $\mathtt{i}(\boldsymbol{x})$. While uniform attribution may be a good baseline, we usually strive for a more informative explanation.

3.3 Explaining Inlierness: Random Features Approach

To overcome the limitations of the approach above, we explore a second approach to explaining inlierness, where the neuralization is based on a feature map representation of the KDE model. For this, we first recall that any kernel-based

model also admits a formulation in terms of the feature map $\Phi(\boldsymbol{x})$ associated to the kernel, i.e. $\mathbb{K}(\boldsymbol{x}, \boldsymbol{x}') = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{x}') \rangle$. In particular Eq.(1) can be equivalently rewritten as:

$$\tilde{p}(\boldsymbol{x}) = \Big\langle \Phi(\boldsymbol{x}), \frac{1}{N} \sum_{k=1}^{N} \Phi(\boldsymbol{u}_k) \Big\rangle, \tag{2}$$

i.e. the product in feature space of the current example and the dataset mean. Here, we first recall that there is no explicit finite-dimensional feature map associated to the Gaussian kernel. However, such feature map can be approximated using the framework of random features [36]. In particular, for a Gaussian kernel, features can be sampled as

$$\widehat{\Phi}(\boldsymbol{x}) = \frac{\sqrt{2}}{H} \big(\cos(\boldsymbol{\omega}_j^\top \boldsymbol{x} + b_j) \big)_{j=1}^{H}, \tag{3}$$

with $\boldsymbol{\omega}_j \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I)$ and $b_j \sim \mathcal{U}(0, 2\pi)$, and where the mean and scale parameters of the Gaussian are $\boldsymbol{\mu} = \mathbf{0}$ and $\sigma = \sqrt{2\gamma}$. The dot product $\langle \widehat{\Phi}(\boldsymbol{x}), \widehat{\Phi}(\boldsymbol{x}') \rangle$ converges to the Gaussian kernel as more and more features are being drawn. In practice, we settle for a fixed number H of features. Injecting the random features in Eq. (2) yields the two-layer architecture:

$$\begin{aligned} h_j &= \sqrt{2} \cos\left(\boldsymbol{\omega}_j^\top \boldsymbol{x} + b_j\right) \cdot \mu_j && \text{(layer 1)} \\ \widehat{\mathbb{i}}(\boldsymbol{x}) &= \tfrac{1}{H} \textstyle\sum_{j=1}^{H} h_j && \text{(layer 2)} \end{aligned}$$

where $\mu_j = \frac{1}{N} \sum_{k=1}^{N} \sqrt{2} \cos(\boldsymbol{\omega}_j^\top \boldsymbol{u}_k + b_j)$ and with $(\boldsymbol{\omega}_j, b_j)_j$ drawn from the distribution given above. This architecture produces at its output an approximation of the true inlierness score $\mathbb{i}(\boldsymbol{x})$ which becomes increasingly accurate as H becomes large. Here, the first layer is a detection layer with a cosine nonlinearity, and the second layer performs average pooling. The structure of the neural network computation is illustrated on our one-dimensional example in Fig. 3.

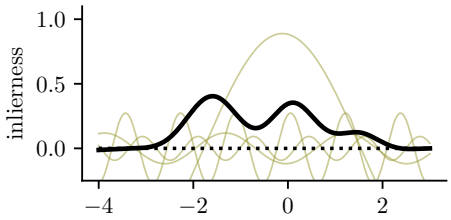

Fig. 3. Kernel density estimation approximated with random features (four of them are depicted in the figure). Unlike the Gaussian kernel, random features have a clear directionality in input space, thereby enabling a feature-wise explanation.

This structure of the inlierness computation is more amenable to explanation. In the top layer, the pooling operation can be attributed based on the summands. In order words, we can apply

$$R_j = \frac{h_j}{\sum_j h_j} \widehat{\mathbb{i}}(\boldsymbol{x})$$

for the first step of redistribution of $\widehat{\mathbb{i}}(\boldsymbol{x})$. More importantly, in the first layer, the random features have now a clear directionality (given by the vectors $(\boldsymbol{\omega}_j)_j$), which we can use for attribution on the input features. In particular, we can apply the propagation rule:

$$R_i = \sum_j \frac{[\boldsymbol{\omega}_j]_i^2}{\|\boldsymbol{\omega}_j\|^2} \cdot R_j.$$

Compared to the direct approach of Section 3.2, the explanation produced here assigns different scores for each input feature. Moreover, while the estimate of inlierness $\widehat{\mathbb{i}}(\boldsymbol{x})$ converges to the true KDE inlierness score $\mathbb{i}(\boldsymbol{x})$ as more random features are being drawn, we observe similar convergence for the *explanation* associated to the inlier prediction.

4 K-Means Clustering

Another important class of unsupervised models is clustering. K-means is a popular algorithm for identifying clusters in the data. The k-means model represents each cluster c with a centroid $\boldsymbol{\mu}_c \in \mathbb{R}^d$ corresponding to the mean of the cluster members. It assigns data onto clusters by first computing the distance between the data point and each cluster, e.g.

$$d_c(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{\mu}_c\| \tag{4}$$

and chooses the cluster with the lowest distance $d_c(\boldsymbol{x})$. Once the data has been clustered, it is often the case that we would like to gain understanding of why a given data point has been assigned to a particular cluster, either for validating a given clustering model or for getting novel insights on the cluster structure of the data.

4.1 Explaining Cluster Assignments

As a starting point for applying our explanation framework, we need to identify a function $f_c(\boldsymbol{x})$ that represents well the assignment onto a particular cluster c, e.g. a function that is larger than zero when the data point is assigned to a given cluster, and less than zero otherwise.

The distance function $d_c(\boldsymbol{x})$ on which the clustering algorithm is based is however not directly suitable for the purpose of explanation. Indeed, $d_c(\boldsymbol{x})$ tends to be inversely related to cluster membership, and it also does not take into account how far the data point is from other clusters. In [18], it is proposed to contrast

the assigned cluster with the competing clusters. In particular, k-means cluster membership can be modeled as the difference of (squared) distances between the nearest competing cluster and the assigned cluster c:

$$f_c(\boldsymbol{x}) = \min_{k \neq c} \left\{ d_k^2(\boldsymbol{x}) \right\} - d_c^2(\boldsymbol{x}) \tag{5}$$

The paper [18] shows that this contrastive strategy results in a two-layer neural network. In particular, Eq. (5) can be rewritten as the two-layer neural network:

$$\begin{aligned} h_k &= \boldsymbol{w}_k^\top \boldsymbol{x} + b_k && \text{(layer 1)} \\ f_c(\boldsymbol{x}) &= \min_{k \neq c} \left\{ h_k \right\} && \text{(layer 2)} \end{aligned}$$

where $\boldsymbol{w}_k = 2(\boldsymbol{\mu}_c - \boldsymbol{\mu}_k)$ and $b_k = \|\boldsymbol{\mu}_k\|^2 - \|\boldsymbol{\mu}_c\|^2$. The first layer is a linear layer that depends on the centroid locations and provides a clear directionality in input space. The second layer is a hard min-pooling. Once the neural network structure of cluster membership has been extracted, we can proceed with explanation techniques such as LRP by first reverse-propagating cluster evidence in the top layer (contrasting the given cluster with all cluster competitors) and then further propagating in the layer below. In particular, we first apply the soft argmin redistribution

$$R_k = \frac{\exp(-\beta h_k)}{\sum_{k \neq c} \exp(-\beta h_k)} \cdot f_c(\boldsymbol{x})$$

where β is a hyperparameter to be selected. An advantage of the soft argmin over its hard counterpart is that this does not create an abrupt transition between nearest competing clusters, which would in turn cause nearly identical data points with the same cluster decision to result in a substantially different explanation. Finally, the last step of redistribution on the input features can be achieved by leveraging the orientation of linear functions in the first layer, and applying the redistribution rule:

$$R_i = \sum_{k \neq c} \frac{[\boldsymbol{w}_k]_i^2}{\|\boldsymbol{w}_k\|^2} R_k.$$

Overall, these two redistribution steps provide us with a way of meaningfully attributing the cluster evidence onto the input features.

5 Experiments

We showcase the neuralization approaches presented above on two examples with two types of data: standard vector data representing wholesale customer spending behavior, and image data, more specifically, industrial inspection and scene images.

5.1 Wholesale Customer Analysis

Our first use case is the analysis of a wholesale customer dataset [11]. The dataset consists of 440 instances representing different customers, and for each instance, the annual consumption of the customer in monetary units (m.u.) for the categories 'fresh', 'milk', 'grocery', 'frozen', 'detergents/paper', 'delicatessen' is given. Two additional geographic features are also part of this dataset, however we do not include them in our experiment. We will place our focus on two particular data points with feature values shown in the table below:

Table 1. Excerpt of the Wholesale Customer Dataset [11] where we show feature values, expressed in monetary units (m.u.), for two instances as well as the average values over the whole dataset.

index	fresh	milk	grocery	frozen	detergents / paper	delicatessen
338	9351 m.u.	1347 m.u.	2611 m.u.	8170 m.u.	442 m.u.	868 m.u.
339	3 m.u.	333 m.u.	7201 m.u.	15601 m.u.	15 m.u.	550 m.u.
AVG	12000 m.u.	5796 m.u.	7951 m.u.	3072 m.u.	2881 m.u.	1525 m.u.

Instance 338 has rather typical levels of spending across categories, in general slightly lower than average, but with high spending on frozen products. Instance 339 has more extreme spending with almost no spending on fresh products and detergents and very high spending on frozen products.

To get further insights into the data, we construct a KDE model on the whole data and apply our analysis to the selected instances. Each input feature is first mapped to the logarithm and standardized (mean 0 and variance 1). We choose the kernel parameter $\gamma = 1$. We use a leave-one-out approach where the data used to build the KDE model is the whole data except the instance to be predicted and analyzed. The number of random features is set to $H = 2500$ such that the computational complexity of the inlier model stays within one order of magnitude to the original kernel model. Predictions on the whole dataset and analysis for the selected instances is shown in Fig. 4.

Instance 338 is predicted to be an inlier, which is consistent with our initial observation that the levels of spending across categories are on the lower end but remain usual. We can characterize this instance as a typical small customer. We also note that the feature 'frozen' contributes less to inlierness according to our analysis, probably due to the spending on that category being unusually high for a typical small customer.

Instance 339 has an inlierness score almost zero, which is consistent with the observation in Table 1 that spending behavior is extremal for multiple product categories. The decomposition of an inlierness score of almost zero on the different categories is rather uninformative, hence, for this customer, we look at what explains outlierness (bottom of Fig. 4). We observe as expected that categories where spending behavior diverges for this instance are indeed strongly represented in the explanation of outlierness, with 'fresh', 'milk', 'frozen' and

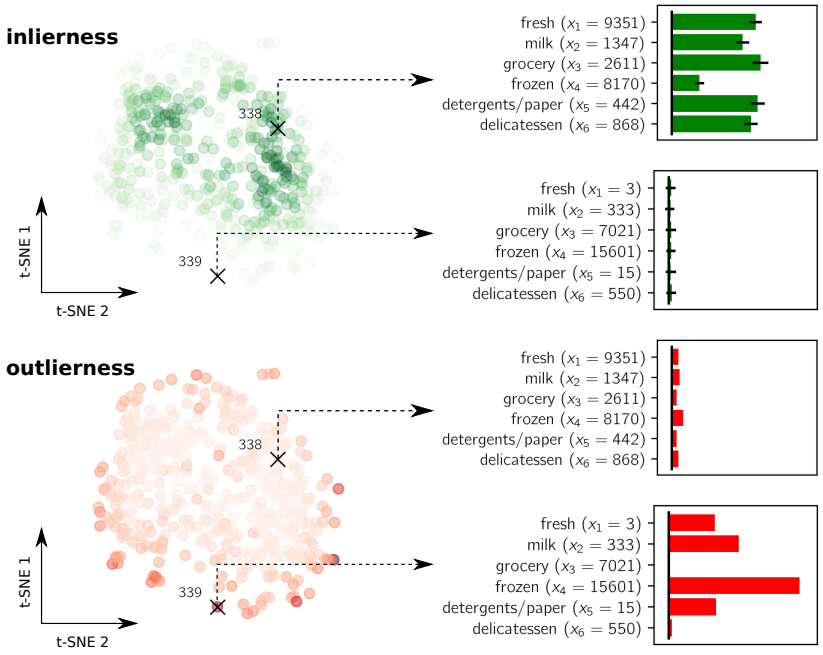

Fig. 4. Explanation of different predictions on the Wholesale Customers Dataset. The dataset is represented on the left as a t-SNE plot (perplexity 100) and each data point is color-coded according to its predicted inlierness and outlierness. On the right, explanation of inlierness and outlierness in terms of input features for two selected instances. Large bars in the plot correspond to strongly contributing features. For explanation of inlierness, error bars are computed over 100 trials of newly drawn random features.

'detergents/paper' contributing almost all evidence for outlierness. Surprisingly, we observe that extremely low spending on 'fresh' is underrepresented in the outlierness score, compared to other categories such as 'milk' or 'frozen' where spending is less extreme. This apparent contradiction will be resolved by a cluster analysis.

Using the same logarithmic mapping and standardization step as for the KDE model, we now train a k-means model on the data and set the number of clusters to 6. Training is repeated 10 times with different centroid initializations, and we retain the model that has reached the lowest k-means objective. The outcome of the clustering is shown in Fig. 5 (left).

We observe that Instance 338 falls somewhere at the border between the green and red clusters, whereas Instance 339 is well into the yellow cluster at the bottom. The decomposition of cluster evidence for these two instances is shown on the right. Because Instance 338 is at the border between two clusters, there is no evidence of membership to one or another cluster, and the decomposition of such (lack of) evidence results in an explanation that is zero for all categories.

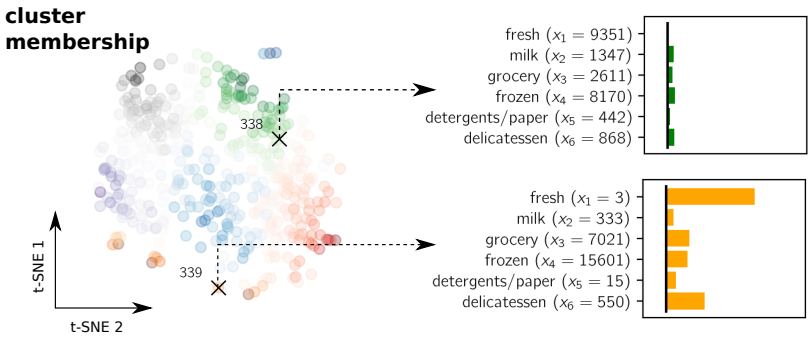

Fig. 5. On the left, a t-SNE representation of the Wholesale Customers Dataset, color-coded by cluster membership according to our k-means model, and where opacity represents evidence for the assigned cluster, i.e. how deep into its cluster the data point is. On the right, explanation of cluster assignments for two selected instances.

The decomposition of the cluster evidence for Instance 339, however, reveals that its cluster membership is mainly due to a singular spending pattern on the category 'fresh'. To shed further light into this decision, we look at the cluster to which this instance has been assigned, in particular, the average spending of cluster members on each category. This information is shown in Table 2.

Table 2. Average spending per category in the cluster to which Instance 339 has been assigned.

cluster	fresh	milk	grocery	frozen	detergents / paper	delicatessen
Yellow	616 m.u.	3176 m.u.	6965 m.u.	1523 m.u.	1414 m.u.	135 m.u.

We observe that this cluster is characterized by low spending on fresh products and delicatessen. It may be a cluster of small retailers that, unlike supermarkets, do not have substantial refrigeration capacity. Hence, the very low level of spending of Instance 339 on 'fresh' products puts it well into that cluster, and it also explains why the outlierness of Instance 339 is not attributed to 'fresh' but to other features (cf. Fig. 4). In particular, what distinguishes Instance 339 from its cluster is a very high level of spending on frozen products, and this is also the category that contributes the most to outlierness of this instance according to our analysis of the KDE model.

Traditionally, cluster membership has been characterized by more basic approaches such as population statistics of individual features (e.g. [8]). Fig. 6 shows such analysis for Instances 338 and 339 of the Wholesale Customer Dataset. Although similar observations to the ones above can be made from this simple statistical analysis, e.g. the feature 'frozen' appears to contradict the membership of Instance 339 to Cluster 4, it is not clear from this simple analysis what

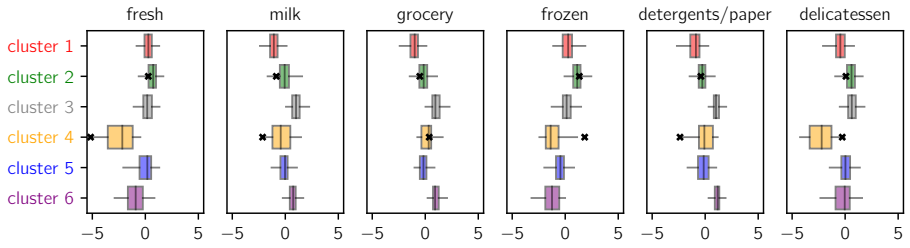

Fig. 6. Population statistics of individual features for the 6 clusters. The black cross in Cluster 2 is Instance 338, the black cross in Cluster 4 is Instance 339. Features are mapped to the logarithm and standardized.

makes Instance 339 a member of Cluster 4 in the first place. For example, while the feature 'grocery' of Instance 339 is within the inter quartile range (IQR) of Cluster 4 and can therefore be considered typical of that cluster, other clusters have similar IQRs for that feature. Moreover, Instance 339 falls significantly outside Cluster 4's IQR for other features. In comparison, our LRP approach more directly and reliably explains the cluster membership and outlierness of the considered instances. Furthermore, population statistics of individual features may be misleading on non-linear models (such as kernel clustering) and does not scale to high-dimensional data, such as image data.

Overall, our analysis allows to identify on a single-instance basis features that contribute to various properties relating this instance to the rest of the data, such as inlierness/outlierness and cluster membership. As our analysis has revealed, the insights that are obtained go well beyond a traditional data analysis based on looking at population statistics for individual features, or a simple inspection of unsupervised learning outcomes.

5.2 Image Analysis

Our next experiment looks at explanation of inlierness, outlierness, and cluster membership for image data. Unlike the example above, relevant image statistics are better expressed at a more abstract level than directly on the pixels. A popular approach consists of using a pretrained neural model (e.g. the VGG-16 network [46]), and use the activations produced at a certain layer as input.

We first consider the problem of anomaly detection for industrial inspection and use for this an image of the MVTec AD dataset [6], specifically, an image of wood where an anomalous horizontal scratch can be observed. The image is shown in Fig. 7 (left). We feed that image to a pretrained VGG-16 network and collect the activations at the output of Block 5 (i.e. at the output of the feature extractor). We consider each spatial location at the output of that block as a data point and build a KDE model (with $\gamma = 0.05$) on the resulting dataset. We then apply our analysis to attribute the predicted inlierness/outlierness to

the activations of Block 5. In practice, we need to consider the fact that any attribution on a deactivated neuron cannot be redistributed further to input pixels as there is no pattern in pixel space to attach to. Hence, the propagation procedure must be carefully implemented to address this constraint, possibly by only redistributing a limited share of the model output. The details are given in Appendix A. As a last step, we take relevance scores computed at the output of Block 5 and pursue the relevance propagation procedure in the VGG-16 network using standard LRP rules until the pixels are reached. Explanations obtained for inlierness and outlierness of the wood image of interest are shown in Fig. 7.

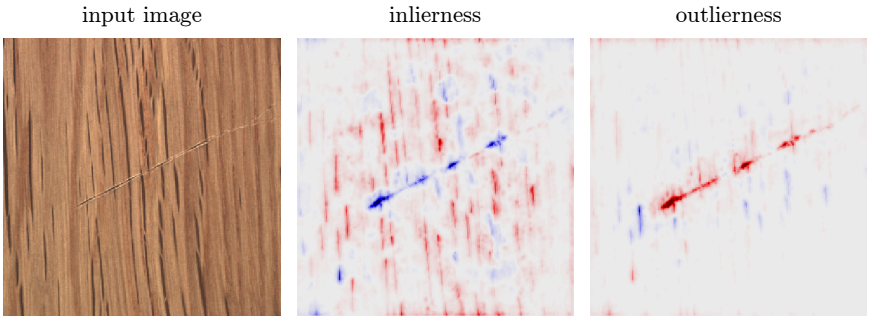

Fig. 7. Exemplary image from the MVTec AD dataset along with the explanation of an inlier/outlier prediction of a KDE model built at the output of the VGG-16 feature extractor. Red color indicates positively contributing pixels, blue color indicates negatively contributing pixels, and gray indicates irrelevant pixels.

It can be observed that pixels associated to regular wood stripes are the main contributors to inlierness. Instead, the horizontal scratch on the wood panel is a contributing factor for outlierness. Hence, with our explanation method, we can precisely identify, on a pixel-wise basis what are the factors that contribute for/against predicted inlierness and outlierness.

We now consider some image of the SUN 2010 database [52], an indoor scene containing different pieces of furniture and home appliances. We consider the same VGG-16 network as in the experiment above and build a dataset by collecting activations at each spatial location of the output of Block 5. We then apply the k-means algorithm on this dataset with the number of clusters hard-coded to 5. Once the clustering model has been built, we rescale each cluster centroid to fixed norm. We then apply our analysis attribute the cluster membership scores to the activations at the output of Block 5. As for the industrial inspection example above, we must adjust the LRP rules so that deactivated neurons are not attributed relevance. The details of the LRP procedure are given in Appendix A. Obtained relevance scores are then propagated further to the input pixels using standard LRP rules. Resulting explanations are shown in Fig. 8.

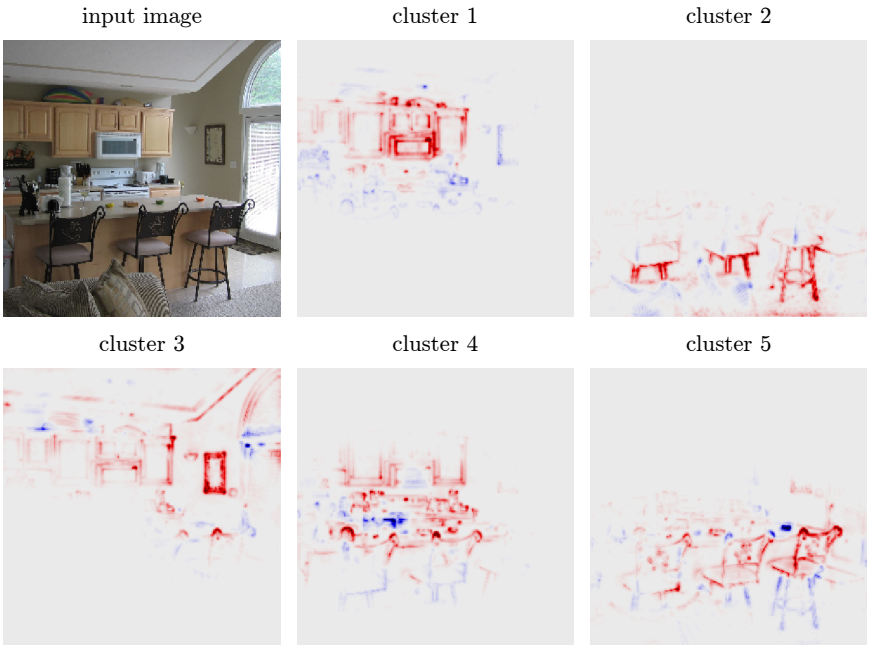

Fig. 8. Exemplary image and explanation of cluster assignments of a k-means model built at the output of the VGG-16 feature extractor. Red, blue and gray indicate positively contributing, negatively contributing, and irrelevant pixels respectively.

We observe that different clusters identify distinct concepts. For example, one cluster focuses on the microwave oven and the surrounding cupboards, a second cluster represents the bottom part of the bar chairs, a third cluster captures the kitchen's background with a particular focus on a painting on the wall, the fourth cluster captures various objects on the table and in the background, and a last cluster focuses on the top-part of the chairs. While the clustering representation extracts distinct human-recognizable image features, it also shows some limits of the given representation, for example, the concept 'bar chair' is split in two distinct concepts (the bottom and top part of the chair respectively), whereas the clutter attached to Cluster 4 is not fully disentangled from the surrounding chairs and cupboards.

Overall, our experiments on image data demonstrate that neuralization of unsupervised learning models can be naturally integrated with existing procedures for explaining deep neural networks. This enables an application of our method to a broad range of practical problems where unsupervised modeling is better tackled at a certain level of abstraction and not directly in input space.

6 Conclusion and Outlook

In this paper, we have considered the problem of explaining the predictions of unsupervised models, in particular, we have reviewed and extended the neuralization / propagation approach of [19, 18] which consists of rewriting, without retraining, the unsupervised model as a functionally equivalent neural network, and applying LRP in a second step. On two models of interest, kernel density estimation and k-means, we have highlighted a variety of techniques that can be used for neuralization. This includes the identification of log-mean-exp pooling structures, the use of random features, and the transformation of a difference of (squared) distances into a linear layer. The capacity of our approach to deliver meaningful explanations was highlighted on two examples covering simple tabular data and images including their mapping on some layer of a convolutional network.

While our approach delivers good quality explanations at low computational cost, there are however still a number of open questions that remain to be addressed to further solidify the neuralization-propagation approach, and the explanation of unsupervised models in general.

A first question concerns the applicability of our method to a broader range of practical scenarios. We have highlighted how neuralized models can be built not only in input space but also on some layer of a deep neural network, thereby bringing explanations to much more complex unsupervised models. However, there is a higher diversity of unsupervised learning algorithms that are encountered in practice, including energy-based models [16], spectral methods [44, 33], linkage clustering [12], non-Euclidean methods [27], or prototype-based anomaly detection [14]. An important future work will therefore be to extend the proposed framework to handle this heterogeneity of unsupervised machine learning approaches.

Another question is that of validation. There are many possible LRP propagation rules that one can define in practice, as well as potentially multiple neural network reformulations of the same unsupervised model. This creates a need for reliable techniques to evaluate the quality of different explanation methods. While techniques to evaluate explanation quality have been proposed and successfully applied in the context of supervised learning (e.g. based on feature removal [39]), further care needs to be taken in the unsupervised scenario, in particular, to avoid that the outcome of the evaluation is spuriously affected by such feature removals. As an example, removing some feature responsible for some predicted anomaly may unintentionally cause some new artefact to be created in the data. That would in turn increase the anomaly score instead of lowering it as it was originally intended [19].

In addition to further extending and validating the neuralization-propagation approach, one needs to ask how to develop these explanation techniques beyond their usage as a simple visualization or data exploration tool. For example, it remains to demonstrate whether these explanation techniques, in combination with user feedback, can be used to systematically verify and improve the un-

supervised model at hand (e.g. as recently demonstrated for supervised models [49, 2]). Some initial steps have already been taken in this direction [20, 38].

Acknowledgements

This work was supported by the German Ministry for Education and Research under Grant 01IS14013A-E, Grant 01GQ1115, Grant 01GQ0850, as BIFOLD (ref. 01IS18025A and ref. 01IS18037A) and Patho234 (ref. 031LO207), the European Union's Horizon 2020 programme (grant no. 965221), and the German Research Foundation (DFG) as Math+: Berlin Mathematics Research Center (EXC 2046/1, project-ID: 390685689). This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea Government under Grant 2017-0-00451 (Development of BCI Based Brain and Cognitive Computing Technology for Recognizing User's Intentions using Deep Learning) and Grant 2019-0-00079 (Artificial Intelligence Graduate School Program, Korea University).

A Attribution on CNN Activations

Propagation rules mentioned in Sections 3 and 4 are not suited for identifying relevant neurons at some layer of a neural network when the goal is to propagate the relevance further down the layers of the neural network, e.g. to obtain a pixel-wise explanation. What we need to ensure in such scenario is that all relevant information is expressed in terms of *activated* neurons as they are the only ones for which the associated relevance can be grounded to a specific pattern in the pixel space. One possible approach is to decompose the relevance propagation into a propagating term and a non-propagating (or 'dissipating') one, which leads to a partial (although still useful) explanation. In the following, we describe the approaches we have taken to achieve our extension of explanations to deep models.

A.1 Attributing Outlierness

The activations in the first layer of the neuralized outlier model is

$$h_k = \|\boldsymbol{a} - \boldsymbol{u}_k\|^2$$

and the relevance that arrives on the corresponding neuron is given by $R_k = p_k \text{LME}^{-\gamma}_{k'}\{h_{k'}\}$ with $p_k = \frac{\exp(-\beta h_k)}{\sum_{k'} \exp(-\beta h_{k'})}$. Relevance associated to neuron k can be expressed as:

$$R_k = \underbrace{p_k \cdot \boldsymbol{a}^\top(\boldsymbol{a} - \boldsymbol{u}_k)}_{R^{\text{dot}}_k} + \underbrace{p_k \cdot (\boldsymbol{u}_k^\top(\boldsymbol{u}_k - \boldsymbol{a}) + \text{LME}^{-\gamma}_{k'}\{h_{k'} - h_k\})}_{R^{\text{res}}_k}$$

where we have used the commutativity of the LME function and the distributivity of the squared norm to decompose the relevance in two terms, one that can be meaningfully redistributed on the activations, and one that cannot be redistributed. Redistribution in the first layer can then proceed as:

$$R_i = \sum_k \frac{a_i \cdot (a_i - u_{ik})}{\sum_i a_i \cdot (a_i - u_{ik})} R_k^{\text{dot}}$$

It is easy to demonstrate from this equation that any neuron with $a_i = 0$ (i.e. deactivated) will not be attributed any relevance.

A.2 Attributing Inlierness

Neurons in the first layer of the inlierness model based on random features, have activations given by:

$$h_j = \sqrt{2}\cos(\boldsymbol{\omega}_j^\top \boldsymbol{a} + b_j) \cdot \mu_j$$

and relevance scores $R_j = h_j / H$. Using a simple trigonometric identity, we can rewrite the relevance scores in terms of unphased sine and cosine functions as:

$$R_j = \underbrace{\big(-\sin(\boldsymbol{\omega}_j^\top \boldsymbol{a}) \sin(b_j) \cdot c_j \big)}_{R_j^{\text{sin}}} + \underbrace{\cos(\boldsymbol{\omega}_j^\top \boldsymbol{a}) \cos(b_j) \cdot c_j}_{R_j^{\text{cos}}}$$

where $c_j = \frac{1}{H}\sqrt{2}\mu_j$. We propose the redistribution rule:

$$R_i = \sum_j \frac{a_i \omega_{ij}}{\sum_i a_i \omega_{ij}} R_j^{\text{sin}} + \sum_j \frac{a_i \omega_{ij}}{\epsilon_j + \sum_i a_i \omega_{ij}} R_j^{\text{cos}}$$

where ϵ_j is a term set to be of same sign as the denominator, and that addresses the case where a positive R_j^{cos} comes with a near-zero response $\boldsymbol{\omega}_j^\top \boldsymbol{a}$, by 'dissipating' some of the relevance R_j^{cos}.

A.3 Attributing Cluster Membership

The activation in the first layer of the neuralized cluster membership model is:

$$h_k = \boldsymbol{w}_k^\top \boldsymbol{a} + b_k$$

and the relevance score is given by $R_k = p_k \min_{k' \neq c}\{h_{k'}\}$ with $p_k = \frac{\exp(-\beta h_k)}{\sum_{k'} \exp(-\beta h_{k'})}$. Similar to the outlier case, we decompose the relevance score as:

$$R_k = \underbrace{p_k \cdot \boldsymbol{a}^\top \boldsymbol{w}_k}_{R_k^{\text{dot}}} + \underbrace{p_k \cdot (b_k + \min_{k' \neq c}\{h_{k'} - h_k\})}_{R_k^{\text{res}}}$$

and only consider the first term for propagation. Specifically, we apply the propagation rule:

$$R_i = \sum_k \frac{a_i w_{ik}}{\sum_i a_i w_{ik}} R_k^{\text{dot}}$$

where it can again be shown that only activated neurons are attributed relevance.

References

1. M. Alber, S. Lapuschkin, P. Seegerer, M. Hägele, K. T. Schütt, G. Montavon, W. Samek, K.-R. Müller, S. Dähne, and P. Kindermans. iNNvestigate neural networks! *J. Mach. Learn. Res.*, 20:93:1–93:8, 2019.
2. C. J. Anders, L. Weber, D. Neumann, W. Samek, K.-R. Müller, and S. Lapuschkin. Finding and removing Clever Hans: Using explanation methods to debug and improve deep models. *Information Fusion*, 77:261–295, 2022.
3. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 07 2015.
4. D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Müller. How to explain individual classification decisions. *J. Mach. Learn. Res.*, 11:1803–1831, 2010.
5. D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
6. P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *Int. J. Comput. Vis.*, 129(4):1038–1059, 2021.
7. A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997.
8. P. Chapfuwa, C. Li, N. Mehta, L. Carin, and R. Henao. Survival cluster analysis. In M. Ghassemi, editor, *ACM Conference on Health, Inference, and Learning*, pages 60–68. ACM, 2020.
9. G. Ciriello, M. L. Miller, B. A. Aksoy, Y. Senbabaoglu, N. Schultz, and C. Sander. Emerging landscape of oncogenic signatures across human cancers. *Nature Genetics*, 45(10):1127–1133, Sept. 2013.
10. M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *NIPS*, pages 24–30. MIT Press, 1995.
11. N. G. C. F. M. de Abreu. Análise do perfil do cliente recheio e desenvolvimento de um sistema promocional. Master's thesis, Instituto Universitário de Lisboa, 2011.
12. J. C. Gower and G. J. S. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, 18(1):54, 1969.
13. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5):93:1–93:42, 2019.
14. S. Harmeling, G. Dornhege, D. Tax, F. Meinecke, and K.-R. Müller. From outliers to prototypes: ordering data. *Neurocomputing*, 69(13-15):1608–1618, 2006.
15. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.
16. G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, 2002.
17. A. K. Kau, Y. E. Tang, and S. Ghose. Typology of online shoppers. *Journal of Consumer Marketing*, 20(2):139–156, Apr. 2003.
18. J. R. Kauffmann, M. Esders, G. Montavon, W. Samek, and K.-R. Müller. From clustering to cluster explanations via neural networks. *CoRR*, abs/1906.07633, 2019.
19. J. R. Kauffmann, K.-R. Müller, and G. Montavon. Towards explaining anomalies: A deep Taylor decomposition of one-class models. *Pattern Recognit.*, 101:107198, 2020.

20. J. R. Kauffmann, L. Ruff, G. Montavon, and K.-R. Müller. The Clever Hans effect in anomaly detection. *CoRR*, abs/2006.10609, 2020.
21. J. Kim and C. D. Scott. Robust kernel density estimation. *J. Mach. Learn. Res.*, 13:2529–2565, 2012.
22. Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
23. A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.
24. S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications*, 10(1096), 2019.
25. P. Laskov, K. Rieck, C. Schäfer, and K.-R. Müller. Visualization of anomaly detection using prediction sensitivity. In *Sicherheit*, volume P-62 of *LNI*, pages 197–208. GI, 2005.
26. L. J. Latecki, A. Lazarevic, and D. Pokrajac. Outlier detection with kernel density functions. In *MLDM*, volume 4571 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2007.
27. F. T. Liu, K. M. Ting, and Z. Zhou. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 413–422. IEEE Computer Society, 2008.
28. N. Liu, D. Shin, and X. Hu. Contextual outlier interpretation. In *IJCAI*, pages 2461–2467. ijcai.org, 2018.
29. S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774, 2017.
30. B. Micenková, R. T. Ng, X. Dang, and I. Assent. Explaining outliers by subspace separability. In *ICDM*, pages 518–527. IEEE Computer Society, 2013.
31. G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller. Layer-wise relevance propagation: An overview. In W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, editors, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *LNCS*, pages 193–209. Springer, 2019.
32. G. F. Montúfar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, pages 2924–2932, 2014.
33. A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856. MIT Press, 2001.
34. A. M. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *NIPS*, pages 3387–3395, 2016.
35. E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.
36. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, page 1177–1184, 2007.
37. M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016.
38. L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller. A unifying review of deep and shallow anomaly detection. *Proc. IEEE*, 109(5):756–795, 2021.
39. W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Trans. Neural Networks Learn. Syst.*, 28(11):2660–2673, 2017.

40. W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, and K.-R. Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proc. IEEE*, 109(3):247–278, 2021.
41. W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, editors. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer, 2019.
42. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359, 2020.
43. L. S. Shapley. 17. a value for n-person games. In *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 1953.
44. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.
45. K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (Workshop Poster)*, 2014.
46. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
47. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR (Workshop)*, 2015.
48. E. Strumbelj and I. Kononenko. An efficient explanation of individual classifications using game theory. *J. Mach. Learn. Res.*, 11:1–18, 2010.
49. J. Sun, S. Lapuschkin, W. Samek, and A. Binder. Explain and improve: Lrp-inference fine tuning for image captioning models. *Information Fusion*, 77:233–246, 2022.
50. M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 2017.
51. U. von Luxburg, R. C. Williamson, and I. Guyon. Clustering: Science or art? In *ICML Unsupervised and Transfer Learning*, volume 27 of *JMLR Proceedings*, pages 65–80. JMLR.org, 2012.
52. J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN database: Exploring a large collection of scene categories. *Int. J. Comput. Vis.*, 119(1):3–22, 2016.
53. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV (1)*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014.
54. L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *ICLR (Poster)*. OpenReview.net, 2017.