Implementation project
Lab Machine Learning for Data Science
Summer Semester 2023

G. Montavon
Institute of Computer Science
Department of Mathematics and Computer Science
Freie Universität Berlin

Lab ML for Data Science: Part I
Getting Insights into an Unsupervised Dataset

The goal of Part I of the project is to extract insights from a purely unsupervised dataset. We will focus on the following dataset:

> UCI Wholesale customers dataset
> `https://archive.ics.uci.edu/ml/datasets/Wholesale+customers`

This dataset collects the annual spending on different product categories by a collection of wholesale customers located in Portugal. Specifically, we are interested in identifying general or specific patterns about wholesale customers, for example, instances with anomalous spending behavior, or clusters of similarly behaving wholesale customers. In particular, we would like to leverage unsupervised ML techniques to identify anomalies and clusters.

In contrast to a pure prediction approach, the goal will be to go beyond the mere detection of anomalies and to provide in addition an explanation, for example, why specific instances are predicted to be anomalous. Often, the explanation is of greater interest than the prediction itself because it identifies the causes of the anomalies and allows meaningful action to be taken to address these causes.

Additionally, it is also important in a data science context, that the outcomes of the analysis are reproducible. For example, a lab that does not possess exactly the same data should ideally arrive at the same conclusion if it follows roughly the same methodology. To ensure the reproducibility of the outcomes, we will consider introducing mechanisms in the ML approach that favor robustness to resampling.

1 Loading the Data, Preprocessing, Initial Data Analysis

The first step will consist of downloading the dataset and converting it into numerical tables (e.g. numpy arrays). In practice, raw data is rarely directly usable for extracting meaningful insights. In particular, there may be substantial heterogeneity between the different input features. Some features may be physical measurements, monetary measurements, while others may be category indicators or even non-numerical data such as text or images. Hence, a preliminary filtering of what is interesting for the analysis we would like to conduct is desirable.

In the context of the UCI wholesale dataset, one may, for example, want to focus your analysis on the numerical data (i.e. annual spending per category) and drop meta-data such as Channel and the Region indicators. Once such preliminary step has been taken, one has a standard dataset of size $N \times d$ where N is the number of instances (wholesale customers), d is the number of spending categories, and each value in the table can be expressed in monetary unit.

To verify the range and distribution of these values, we can generate some basic statistical visualizations of the data, such as histograms showing for each category the distribution of spendings, or scatter plots showing the correlation between different product categories. A common observation from such basic statistical analysis is that the distributions are heavy tailed, with many instances having rather small spendings, whereas a few may have spendings one or two orders of magnitude above. Any anomaly detection algorithm would systematically highlight those high spenders as anomalous and not make a distinction between spending little and not spending at all.

To address this issue, it can be useful to apply some nonlinear transformation to the data, for example, applying the log function to the features so that the distribution becomes compressed for large values and expanded for small values.

$$x \leftarrow \log(x+1)$$

Here, we add the offset '1' in the logarithm so that zero spending gets mapped to a specific value and not zero. To verify the effect of the transformation, you can recompute the histograms and scatter plots in transformed space and check visually whether the transformation had the desired result.

2 Detecting Anomalies

We now focus on a question of interest, specifically, whether there are anomalous instances in the given data. A basic form of anomaly detection is to verify for each data point whether it has neighbors. A simple measure of anomaly can then be given by the distance (or squared distance) to the nearest neighbor. For example, given some test point with index j in the dataset, one can get this outlier score by performing a minimum over the $N-1$ remaining points in the data, i.e.:

$$\begin{aligned} z_{jk} &= \|\boldsymbol{x}_j - \boldsymbol{x}_k\|^2 \\ y_j &= \min_{k \neq j} z_{jk} \end{aligned} \tag{1}$$

With such anomaly scores, data points can now be ranked from most to least anomalous. For example, one may now be able to extract a top-10 list of the most anomalous instances in the dataset, and print for these 10 instances their recorded spending across categories.

In practice, however, Eq. (1) may not identify top outliers in a way that is sufficiently *reproducible*. Imagine, for example, a point $\boldsymbol{x}_j$ whose nearest neighbor, call it $\boldsymbol{x}_i$ is at a distance of 1 but the second nearest neighbor is at a distance of 10. If the nearest point $\boldsymbol{x}_i$ was not in the dataset (e.g. due to slight variations in the data collection process), the outlierness score of $\boldsymbol{x}_j$ would have changed drastically. An efficient way of addressing this problem is to reconsider the notion of outlierness by considering a point to be an outlier based on multiple neighbors. In other words, a point may still be an outlier even if (by accident) some other point in the data shares roughly the same values.

Such a redefined notion of outlierness can be implemented by replacing the hard minimum in the original equation by a soft minimum, i.e.:

$$y_j = \operatorname{soft}\min_{k \neq j} \left\{ z_{jk} \right\} \tag{2}$$

with

$$\operatorname{soft}\min_{k\neq j}\{z_{jk}\} = -\frac{1}{\gamma}\log\frac{1}{N-1}\sum_{k\neq j}\exp(-\gamma z_{jk}).$$

The softmin can be interpreted as a generalized F-mean with $F(t) = \exp(-\gamma t)$. The output of such function is also related to the log-likelihood prediced by a kernel density estimator of the rest of the data (see e.g. [2]).

To verify the gain in terms of reproducibility one can gain from using soft min-pooling, one can apply the bootstrap estimator method in statistics, which consists of simulating multiple variants of the same datasets by randomly sampling half of the instances from the original dataset. The anomaly scores can then be computed for each variant of the dataset, and the spread of anomaly scores for each instance can be measured. Such a bootstrapping analysis can be performed for different parameters γ in order to assess which value of γ is needed to achieve the desired robustness. In practice, a small value of γ may lead to more robust estimates; however, setting γ too small may introduce an unacceptable bias on the true outlierness, as the estimator would then depend uniformly on all instances of the data, regardless of their distance from the example of interest. Hence, an appropriate tradeoff should be reached.

3 Explaining Anomalies

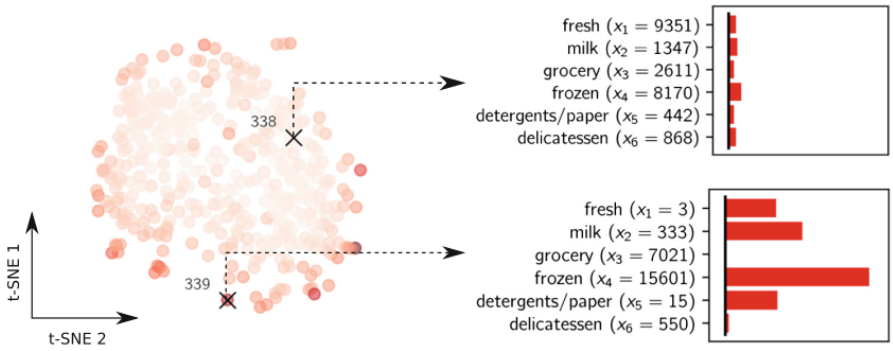
In a data science setting, detecting anomalies can be of limited interest. While it helps the scientist answer the basic question of whether there are anomalies in the data and how frequent they are, one may wish for more insights, for example, whether input features (i.e. spending categories) contribute predominantly to instances being anomalous. Additionally, one might want to extract such sources of anomaly for each anomalous instance in the data. Explainable AI is a set of techniques that allow for such insights. Using the method of Layer-wise Relevance Propagation (see [1] for an overview), attribution of anomaly scores to the input features can be achieved in two steps: First, one identifies to what extent each data point has contributed to the anomaly score of instance j:

$$R_k^{(j)} = \frac{\exp(-\gamma z_k)}{\sum_{k\neq j}\exp(-\gamma z_k)}\cdot y_j \tag{3}$$

Then, these scores can be propagated back to the input features by observing that the (squared) Euclidean distance entering the anomaly score can be decomposed in terms of individual components:

$$R_i^{(j)} = \sum_{k\neq j}\frac{[\boldsymbol{x}_k - \boldsymbol{x}_j]_i^2}{\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^2}\cdot R_k^{(j)} \tag{4}$$

Implementing the attribution process described in Equations (3) and (4), and applying it to each data point enables us to build for each data point a histogram of the type one can see in the image below:



Note that Eqs. (3) and (4) embed the conservation property $\sum_{i=1}^{d} R_i^{(j)} = y_j$. This allows for interpreting each produced score as the share of the anomaly score that is explained by the associated input feature. For technical purposes, it can also serve as a unit test to verify the correct implementation of these rules. As mentioned in Section 2, an important aspect of a data science experiment is to ensure that outcomes are reproducible to a sufficient extent. While we have tested the reproducibility of outlier scores, a similar reproducibility experiment can be performed for the explanations of these outlier scores.

4 Cluster Analysis

Once the strongest anomalies detected in the data have been removed, there are potentially interesting structures that can be retrieved from the non-anomalous part of the data, such as principal components or clusters. Here, we focus on producing a clustering of the remaining data.

A well-known clustering algorithm is k-means, which, like the anomaly detection method above, also relies on the Euclidean distance in input space but this time to perform cluster assignments. The k-means algorithm searches for a set of clusters $(\mathcal{C}_k)_{k=1}^{K}$ and a set of cluster centroids $(\boldsymbol{\mu}_k)_{k=1}^{K}$ that minimize the overall distance between points and their assigned cluster centroid:

$$\min \sum_{k=1}^{K} \sum_{i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$

The centroids $(\boldsymbol{\mu}_k)_{k=1}^{K}$ and cluster assignments $(\mathcal{C}_k)_{k=1}^{K}$ are optimized in an alternate fashion. The optimization problem is non-convex, hence, choosing appropriate initializations for the algorithm as well as repeating the algorithm multiple times in order to reach a lower value of the objective are common practices. An implementation of k-means with such heuristics is readily available in scikit-learn.

In practice, it is also required to set the hyperparameter K which represents the number of clusters in a meaningful way. A common approach is the *elbow method*, which consists of plotting the objective value (corresponding to the spread of the data that cannot be explained by cluster centroids) as a function of the number of clusters K. Looking at this plot, one then tries to identify an inflexion point, or in other words, a value for K such that removing clusters would lead to a severe increase in the error, whereas adding extra clusters would not reduce the error significantly more. The number of clusters K could also be restricted by the user requirement, which could be, for example, to segment the customers into 10 groups at most.

Once an appropriate cluster solution has been found, the entries of a given vector $\boldsymbol{\mu}_k$ can be interpreted as the average spending of cluster members for the different product categories. Hence, it gives insights on what 'typical' wholesale customers look like in the dataset.

References

[1] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller. *Layer-Wise Relevance Propagation: An Overview*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2019.

[2] G. Montavon, J. R. Kauffmann, W. Samek, and K.-R. Müller. *Explaining the Predictions of Unsupervised Learning Models*, volume 13200 of *Lecture Notes in Computer Science*, pages 117–138. Springer, 2020.