Data science for tax administration

Pijnenburg, M.G.F.

6

Extending an Anomaly Detection Benchmark with Auto-encoders, Isolation Forests, and RBMs

In tax administrations and other organizations that apply data science techniques, a frequently heard question is *What algorithm should I use?* In this chapter we make a contribution to answering this question by addressing a class of techniques, *unsupervised anomaly detection algorithms*, that are frequently used in tax administrations. The exact research question is:

> **Research Question** *Unsupervised anomaly detection algorithms play an important role in (tax) fraud detection. What algorithms can be expected to work well under what conditions?*

In this chapter, the recently published benchmark of Goldstein and Uchida [43] for unsupervised anomaly detection is extended with three anomaly detection techniques: Sparse Auto-Encoders, Isolation Forests, and Restricted Boltzmann Machines. The underlying mechanisms of these algorithms differ substantially from the more traditional anomaly detection algorithms, currently present in the benchmark. Results show that in three of the ten data sets, the new algorithms surpass the present collection of 19 algorithms. Moreover, a relation is noted between the nature of the outliers in a data set and the performance of specific (clusters of) anomaly detection algorithms. The chapter is based on the following article:

- M. Pijnenburg and W. Kowalczyk. Extending an anomaly detection benchmark with auto-encoders, isolation forests, and rbms. In *International Conference on Information and Software Technologies*. Springer, 2019. Best Paper Award

6.1 Introduction

The expanding number of anomaly detection algorithms creates the need to compare algorithms objectively. Goldstein and Uchida [43] made a start by providing a benchmark for unsupervised anomaly detection, consisting of 10 data sets and 19 algorithms. In this chapter, we extend the number of algorithms by adding three anomaly detection algorithms: Sparse Auto-encoders, Isolation Forests, and Restricted Boltzmann Machines (RBMs).

The three algorithms are interesting since they have a different underlying mechanism compared to the current algorithms in the benchmark: two of the new algorithms, Sparse Auto-encoders and RBMs, originate from the popular field of (deep) neural networks. Within this field, they are among the simplest and best-known algorithms for detecting anomalies [60]. The third algorithm is based on random trees. The underlying mechanisms differ substantially from the more traditional algorithms in the benchmark that are mostly distance-based, like the k-Nearest Neighbors algorithm and the Local Outlier Factor. Moreover, when auto-encoders and RBMs are applied to anomaly detection, it is usually on image data [103], [102] or sequential data [73], [114]. Hence it is of interest to see their performance on the (small) classical tabular data used in the benchmark [43].

The chapter is organized as follows. In Section 6.2, the three anomaly detection algorithms are described. Then, in Section 6.3, the actual experiments performed are described as well as the data sets of the benchmark. In Section 6.4 we present the results of the experiments and compare these with results mentioned in [43]. The chapter ends with Conclusions and Discussion in Section 6.5. The code used in the experiments is published at [86].

6.2 Theoretical Background

6.2.1 Sparse Auto-encoder

Standard auto-encoders are neural networks with architecture as depicted in Figure 6.1. Sometimes it is required that $W_1 = W_2^t$, for regularization purposes. We will not impose this restriction in this chapter. In its most basic form, as shown in the figure, the network consists of one input layer, a hidden layer, and an output layer with the same number of nodes as the input layer. The network is trained by providing the same observation $\mathbf{v}$ as input *and* output, and requiring to minimize the reconstruction error $\|\text{auto}(\mathbf{v}) - \mathbf{v}\|^2$. Essentially, the network must learn the identity function ('auto' means 'self' in Greek).

The number of nodes in the hidden layer is intentionally limited, such that the en-

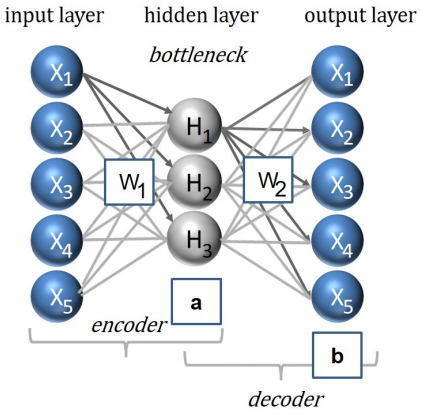

Figure 6.1: Architecture of the simplest form of a standard auto-encoder with one hidden layer. In general, the encoder and the decoder part may consist of more complex neural networks.

coder has to extract the essential information from the input features in order for the decoder to reconstruct the original input as closely as possible. The encoder part of an auto-encoder can thus be seen as a dimensionality-reducing algorithm, reducing the original dimensionality of the input space to the dimensionality of the space formed by the hidden nodes.

Training an auto-encoder is usually done by gradient descent, in particular *stochastic* gradient descent. The latter algorithm speeds up convergence in comparison with standard gradient descent and also introduces some noise that helps to avoid local minima. Backpropagation is generally used to compute the gradient by going backward from the output layer to the input layer.

In the experiments, see Algorithm 1, we used the BFGS (Broyden Fletcher Goldfarb Shanno) algorithm, a quasi-Newton optimization algorithm for minimizing the target function. The BFGS algorithm works well for data sets with a small number of features as are most data sets in the benchmark. The target function in the experiments consists of the reconstruction error – equation (1) in Algorithm 1 –, a standard L_2 regularization term (2) that is added more routinely in training neural networks nowadays, and a sparsity constraint term (3) that is typical for *sparse* auto-encoders.

Sparse auto-encoders are a variant on classical auto-encoders where the 'bottleneck' is not created by limiting the number of hidden nodes, but by requiring that only a limited number of hidden nodes have a high activation value for each observation.

When applying auto-encoders to anomaly detection, at least two approaches may be taken. The first approach uses the dimensionality-reduction characteristic of the

Algorithm 1: Training and Scoring of an auto-encoder with one hidden layer.

Input : A data set $\mathbf{V}$ with p (numeric) features and n observations. We assume each column to be scaled into the range [0,1] using min-max scaling.
$h = \{5, 10, \lfloor p/2 \rfloor, p\}$ the number of hidden nodes,
$\lambda = 0.001$ the learning rate,
$\rho = 0.1$ sparsity hyper-parameter,
$\beta = 0.05$ factor influencing the relative importance of the sparsity term in the cost function,

Output: Reconstruction error of all observations $\mathbf{v} \in \mathbf{V}$.

1 Initialize weight matrix W with random number from a $\mathcal{N}(0, 0.01)$ distribution

2 Call a standard BFGS optimizer for minimizing the target function that consists of the reconstruction error, a regularization term and the sparsity constraint:

3

$$J(W_1, W_2, \mathbf{a}, \mathbf{b}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\|\text{auto}(\mathbf{v}_i) - \mathbf{v}_i\|^2 \quad (6.1)$$

$$+\frac{\lambda}{2}(\|W_1\|^2 + \|W_2\|^2) \quad (6.2)$$

$$+\beta\sum_{i=1}^{n}[\rho\log\frac{\rho}{\widehat{\rho}(\mathbf{v}_i)} + (1-\rho)\log\frac{1-\rho}{1-\widehat{\rho}(\mathbf{v}_i)}], \quad (6.3)$$

where auto($\mathbf{v}$) is the output of the feedforward pass through the network:

$$\text{auto}(\mathbf{v}) = 1/(1+\exp(-W_2 A(\mathbf{v}) - \mathbf{b})), \quad (6.4)$$

$$A(\mathbf{v}) = 1/(1+\exp(-W_1\mathbf{v} - \mathbf{a})), \quad (6.5)$$

and $\widehat{\rho}(\mathbf{v}))$ is the average activation of the nodes in the hidden layer:

$$\widehat{\rho}(\mathbf{v})) = \frac{1}{h}\sum_{i=1}^{h} A_i(\mathbf{v}). \quad (6.6)$$

4 After convergence, pass all observations $\mathbf{v} \in \mathbf{V}$ through the trained auto-encoder and compute the reconstruction error $e(\mathbf{v}) = \|\text{auto}(\mathbf{v}_i) - \mathbf{v}_i\|^2$.

5 **return** the vector of reconstruction errors: $e(\mathbf{v}_1), \ldots, e(\mathbf{v}_n)$.

auto-encoder part. Once trained, the auto-encoder will transform the original features into a new low-dimensional feature space (i.e., the hidden layer). In the new feature space, traditional distance-based anomaly detection techniques may be applied that would not work properly in the original, high-dimensional space due to the 'curse of dimensionality'. This first approach works in particular well for standard auto-encoders. The second approach will work for standard auto-encoders as well as sparse auto-encoders. The idea underlying this approach is that the network will only achieve a low reconstruction error if it focuses on frequently occurring patterns. As a result, observations belonging to infrequent patterns will receive a high reconstruction error. Hence the reconstruction error can serve as an anomaly score. This approach is adopted in this chapter.

6.2.2 Isolation Forest

Isolation forest [71], see Algorithm 5, is an algorithm specifically developed to find anomalies. It resembles the random forest algorithm. As such it is a collection of many trees. However these trees are no decision trees, but 'random trees'. A 'random tree' is a tree where each split involves a randomly selected feature, which is split based on a random value. The underlying assumption of an isolation forest is that anomalies are few and have different values from most observations. As a result, anomalies will often be isolated from the other observations in very few splits. Therefore, by observing the leaf of an observation in many trees, and computing the average distance of these leaves to the root of the trees, anomalies will have a small distance, while normal observations will have a large distance. Hence, the average distance to the root can be used as an anomaly score.

6.2.3 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) is a stochastic neural network with two layers: a *visible layer*, and a *hidden* layer, see Figure 6.2. The network has no output layer like auto-encoders, and signals in the network travel back and forth between the two layers, starting at the visible layer. Both layers are fully connected, i.e., each input node is connected to each hidden node, and vice versa. Moreover the weights of the connections are symmetric $W_{ij} = W_{ji}$. No connections between nodes of the same layer are allowed. All nodes in a *classical* RBM are binary, i.e. can take two values: 0 and 1. This in contrast to auto-encoders. Consequently, numerical inputs have to be discretized and transformed to binary dummy variables. In our experiments, the 'thermometer encoding' is used for the latter, as it preserves the ordering present in numerical features. The difference between thermometer encoding and standard

Algorithm 2: Training and Scoring of Isolation Forest

Input : A data set $\mathbf{V}$ with p (numeric) features. We assume each column to be scaled into the range [0,1] using min-max scaling.
$n\_tree = 100$ the number of trees,
$hlim = 8$ the maximum depth of a single tree,
$n\_samp = 256$ number of observations used in the training sample,
$min\_leaf = 1$ minimum number of observations in a leaf.

Output: Scaled Version of the average path length of each observation $\mathbf{v} \in \mathbf{V}$.

```
Create n_tree random trees:
for i in 1 ... n_tree do
    Take a random sample X ⊂ V of size n_samp
    Initialize first node of tree
    while there is a node N with depth < hlim and # observations > min_leaf do
        randomly select an attribute q
        randomly select a split point s ∈ [0, 1]
        Split node N in 2: q <= p, q > p
    end
end
Compute (scaled version of) Average Path Length, APL(v), for all observations v ∈ V
return the vector of average path lengths: (APL(v_1), ..., APL(v_n)).
```

dummy encode can best be illustrated by the example of representing the value 0.45 of continuous feature with a range $[0, 1]$ that is discretized in ten equal width bins $[0, 0.1), [0.1, 0.2), \ldots$. With standard encoding 0.45 falls in the bin $[0.4, 0.5)]$ and will be representated by the vector $(0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$. With thermometer encoding 0.45 would be represented by the vector $(1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$.

An RBM can be interpreted as a graphical model, or a 'Markov Random Field', see [39]. Consequently, there is a model for the probability distribution over the feature space:

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\text{all feasible } \mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}, \tag{6.7}$$

where Z is a normalization constant (also known as the *partition function*) ensuring that $\sum p(\mathbf{v}) = 1$,

$$Z = \sum_{\substack{\text{all feasible} \\ \mathbf{v},\mathbf{h}}} e^{-E(\mathbf{v},\mathbf{h})}, \tag{6.8}$$

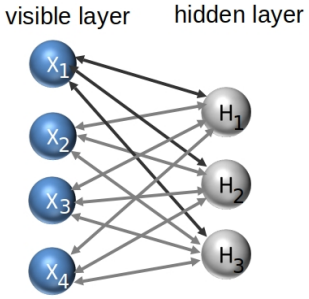

Figure 6.2: Architecture of a Restricted Boltzmann Machine.

and E is the so-called *energy function*,

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} w_{ij} v_i h_j. \tag{6.9}$$

This family of probability distributions is known as 'Boltzmann distributions' and has been subject of study in statistical physics.

Training an RBM amounts to adjusting the parameters of the energy function (6.9) such that the distribution fits the observations of the training data set, see Algorithm 6. Fitting the distribution to observations is done by maximizing the likelihood over the the training set V,

$$\arg\max_{a_i, b_i, w_{ij}} \prod_{\mathbf{v} \in V} p(\mathbf{v}). \tag{6.10}$$

The maximization is usually done by with the help of Contrastive Divergence, see [39], a 'stochastic gradient descent'-like algorithm, showing fast convergence at the cost of approximating the gradient.

After training, the probability of each observation $p(\mathbf{v})$ may be computed by equation (6.7). In practice, however, applying equation (6.7) requires computing the partition function (6.8), which is computationally intractable. Instead, one may notice that $p(\mathbf{v})$ is proportional to,

$$p(\mathbf{v}) \propto e^{-F(\mathbf{v})} = \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})}. \tag{6.11}$$

Here $F(\mathbf{v})$ is the *free energy of an observation* $\mathbf{v}$: the energy that a single configuration would need to have in order to have the same probability as all of the configurations that contain $\mathbf{v}$ [51].

The free energy of an observation can be calculated in linear time, due to the special architecture of a *Restricted* Boltzmann Machine, which does not allow links between hidden nodes, leading to an energy function (6.9) that involves no cross

Algorithm 3: Training and Scoring of an RBM using CD-1

Input : A data set $\mathbf{V}$ with p numeric features, scaled into the range [0,1]
$h = \{5, 10, \lfloor p/2 \rfloor, p\}$ the number of hidden nodes,
$b = \{3, 5, 7, 10\}$ the number of bins for each feature,
$k = 10$ batch size,
$\lambda = 0.01$ initial learning rate,
$m = 0.95$ momentum term (only used for bias vectors)

Output: for each $\mathbf{v} \in \mathbf{V}$: $\exp(-F(\mathbf{v}))$. This expression is proportional to $p(\mathbf{v})$.

Discretize columns of $\mathbf{V}$ into b bins each, using equal width binning followed by thermometer encoding. Denote the new data set with $p \cdot b$ binary columns $\mathbf{V}'$.
Initialize values of matrix W and vectors $\mathbf{a}$ and $\mathbf{b}$ with small uniform random numbers
while *convergence = FALSE* **do**
  randomize order of rows and put rows in batches of size k
  for *each batch* V_0 **do**
    sample binary values H_0 based on values V_0:
    $H_0 = \text{random}_{\{0,1\}}(1/(1 + \exp(-WV_0^t - \mathbf{b})))$
    compute new values V_1 based on H_0: $V_1 = 1/(1 + \exp(-(H_0W)^t - \mathbf{a}))$
    Compute probabilities of hidden nodes H_1 based on V_1 (without sampling): $H_1 = 1/(1 + \exp(-WV_1^t - \mathbf{b}))$
    Adjust weights:
    $W = W + \lambda \cdot (H_0^t V_0 - H_1^t V_1)/k$
    $d\mathbf{a} = d\mathbf{a}_{-1} \cdot m + \lambda \cdot$ column_means $(V_0 - V_1)$
    $\mathbf{a} = \mathbf{a} + d\mathbf{a}$
    $d\mathbf{b} = d\mathbf{b}_{-1} \cdot m + \lambda \cdot$ column_means $(H_0 - H_1)$
    $\mathbf{b} = \mathbf{b} + d\mathbf{b}$
  end
  compute the total free energy $F(\mathbf{V}') = \sum_{\mathbf{v} \in \mathbf{V}'} F(\mathbf{v})$, see equation (6.12)
  Check on convergence:
  if $|(F(\mathbf{V}') - F_{previous}(\mathbf{V}'))/F_{previous}(\mathbf{V}')| < 0.001$ **then**
    convergence = TRUE
  end
  Check on adjustment of λ:
  if $(F(\mathbf{V}') - F_{previous}(\mathbf{V}'))/F_{previous}(\mathbf{V}') > 0.01$ **then**
    $\lambda = 0.1 \cdot \lambda$
  end
end
return for each $\mathbf{v} \in \mathbf{V}'$: $\exp(-F(\mathbf{v}))$, see equation (6.12).

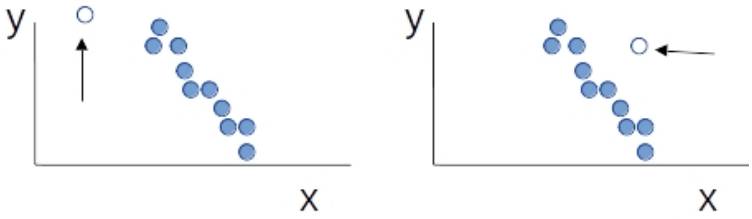

Figure 6.3: 'Univariate' outlier (left, U-index = 0.74) and 'multivariate' outlier (right, U-index = 0.43).

terms like $h_j h_k$. Hence the sum over all possible values of $\mathbf{h}$ in equation (6.11), comes down to summing over each element of $\mathbf{h}$ separately, essentially reducing the time to compute the free energy of an observation from exponential to linear. The free energy of an observation can be expressed most conveniently as,

$$F(\mathbf{v}) = -\sum_{i \in \text{visible}} v_i a_i - \sum_{j \in \text{hidden}} \log(1 + e^{x_j}), \tag{6.12}$$

where $x_j = b_j + \sum_i v_i w_{ij}$ is the input for the hidden node j, see equation (22) of [39] for a derivation.

Applying the procedure above, we obtain for each observation the expression $\exp(-F(\mathbf{v}))$, which is proportional to $p(\mathbf{v})$. Now one may proceed along two lines: (1) if interest lies only in obtaining the k most anomalous cases in a data set, one may order the observations according to $F(\mathbf{v})$ and report the k observations with largest free energy. (2) otherwise, one may obtain a set of outliers by computing the median m and interquartile range IQR of $\exp(-F(\mathbf{v}))$ for all $\mathbf{v} \in V$ and set a threshold $\theta = m - c \cdot IQR$ below which observations are considered outliers.

6.2.4 Type of outliers

A distinction between type of outliers, that will prove fruitful in explaining differences among algorithms in Section 6.4, is that some outliers are multivariate in nature while others are univariate, compare Figure 6.3. In the left subplot a univariate outlier is shown. This outlier can be detected by only looking at one feature (x), while the multivariate outlier of the right subplot requires knowledge about both features.

In practice we do not know a priori the nature of the outliers, as we do not know what observations are outliers. However, in a benchmark situation, we *know* what observations are outliers. The outliers are indicated by a binary feature y taking the value 1 for an outlier and 0 otherwise. In a benchmark situation we can thus quantify

the univariate nature of the outliers by defining a new concept, called the Univariate-index, or *U-index*, as

$$\text{U-index} = \max_{i \in \{1 \ldots p\}} (|\text{corr}(x_i, y)|), \tag{6.13}$$

where corr is the correlation function, p is the number of features of the data set (without the outlier label y). In words, equation (6.13) says to take the maximum of the absolute value of the correlation coefficient of any feature with the binary label indicating the outliers. The index is added to Table 6.1 and will prove useful in Section 6.4.

6.3 Experimental Setup

6.3.1 General

To test the effectiveness of the algorithms mentioned in the previous section, the algorithms are applied to the benchmark data sets of Goldstein and Uchida [43]. The complete code of the experiments can be at [86]. In the experiments we used implementations of the algorithms as can be found in the R-packages: 'autoencoder' (version 1.0) [34], 'IsolationForest' (version 0.0-26) [70], and 'deepnet' (version 0.2) [100]. The latter package is used for the RBM and is adjusted slightly in order to implement a dynamic stopping criterion and the automatic adjustment of the learning rate, see Section 6.3.2.

The set of hyper-parameters that are tested for each algorithm will be explained in Section 6.3.2. For each set of hyper-parameters we will run ten experiments, each time with a different random seed. This will reduce the noise introduced by the random component that is present in all three algorithms. Reported results are averages over all runs. For instance, for an RBM we will test 16 different hyper-parameter settings (4 different number of hidden nodes times 4 different settings for the number of bins). A reported auc in Table 6.2 and Table 6.3 is thus an average over $4 \cdot 4 \cdot 10 = 160$ experiments.

Results are measured using the 'area under the curve' statistic, in line with the approach of Goldstein and Uchida [43].

6.3.2 Setting hyper-parameters

6.3.2.1 Sparse auto-encoder

In the experiments the most basic architecture of a sparse auto-encoder is tested, i.e., with one hidden layer. Since this simple architecture already achieved good res-

ults, see Section 6.4, we have not experimented with more complex variants of auto-encoders.

The main architectural hyper-parameter is the number of hidden nodes h. We choose $h \in \{5, 10, p/2, p\}$, where p is the number of features in the data set, excluding the label indicating the outliers. If $p/2$ is not an integer, we rounded downwards. An exception is made for the 'speech' data set that contains 400 features, requiring an exceptional long run time. For 'speech' we take $h \in \{5, 10, 25, 50\}$. The numbers 5 and 10 hidden nodes have been chosen since some initial experiments indicated that a reasonably low reconstruction error could be obtained with these numbers. The numbers $p/2$ and p are added to ensure that data sets with more features (more possible patterns) have a network with larger expressive power.

The learning rate λ is set to a value of 0.001 for all data sets, since this value ensured a smooth decreasing target function for all data sets. The sparsity parameter ρ is fixed to 0.1 as recommended by Ng in his lecture notes on auto-encoders [80].

6.3.2.2 Isolation forest

The isolation forest algorithm is robust concerning the values of its hyper-parameters. For all hyper-parameters we choose the values as recommended by Liu et al. [71].

6.3.2.3 Restricted Boltzmann Machine

The number of hidden nodes for the RBM is set equal to the values chosen for the sparse auto-encoder, i.e. $h \in \{5, 10, \lfloor p/2 \rfloor, p\}$ and $h \in \{5, 10, 25, 50\}$ for the 'speech' data set.

A classical RBM needs binary input. For this reason we pre-processed the data for RBM's by discretizing each feature into b bins, using equal width binning. Subsequently, a dummy variable is constructed for each bin, using thermometer encoding as mentioned in Section 6.2.3. In the experiments we set $b \in \{3, 5, 7, 10\}$.

The learning rate λ and the stopping criterion of the RBM is set dynamically. Initially $\lambda = 0.01$, subsequently λ is decreased by a factor 10 as soon as free energy of all observations between two epochs rises with more than 1 per cent. The algorithm is stopped as soon as the free energy of the data set changes less than 1 promille. The *free energy* serves as a proxy for the (log-) likelihood (6.10). The log-likelihood can be separated in two terms like,

$$\log \prod_{\mathbf{v} \in V} p(\mathbf{v}) = -\log Z(W, \mathbf{a}, \mathbf{b}) - \sum_{\mathbf{v} \in V} F(\mathbf{v}). \tag{6.14}$$

The last term is the free energy of the data set.

6.3.3 Data Sets

Table 6.1 provides a summary of the data sets of the benchmark of Goldstein and Uchida. [43]. All features in the data sets are numeric. Most data sets are originally posted for classification tasks. To make the data sets suitable for anomaly detection, typically observations from one specific class are labeled anomalies, while all other observations are considered normal cases.

As a data pre-processing step, a min-max scaling is applied to all features x of all data sets, resulting in a range of $[0, 1]$ for each feature,

$$x_{sc} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}. \tag{6.15}$$

Below we will give a short description of each data set.

	data set name	number of rows	number of columns	outliers	percentage outliers	U-index
1	breast cancer	367	30	10	2.72	0.570
2	pen global	809	16	90	11.1	0.600
3	letter	1.600	32	100	6.25	0.193
4	speech	3.686	400	61	1.65	0.079
5	satellite	5.100	36	75	1.49	0.308
6	pen local	6.724	16	10	0.15	0.047
7	annthyroid	6.916	21	250	3.61	0.419
8	shuttle	46.464	9	878	1.89	0.675
9	aloi	50.000	27	1.508	3.02	0.029
10	kdd 1999	620.098	29	1.052	0.17	0.678

Table 6.1: Summary of the data sets used to compare the various anomaly detection algorithms. See equation (6.13) for the definition of the U-index.

6.3.3.1 Breast Cancer

This data set is derived from the Wisconsin Breast Cancer data set that contains medical data of 569 patients. The data consists of features derived from digitized images of breast mass, obtained via a Fine Needle Aspirate. In the original data set there are 357 patients with benign breast cancer and 212 patients with malignant breast cancer. In the data set prepared for anomaly detection, all benign patients are kept, while the first 10 patients with malignant breast cancer are labeled as anomalies.

6.3.3.2 Pen Global

The observations in this data set are feature vectors derived from images of the digit '8', handwritten several times by 44 different writers. The feature vectors have a length of 16 and contain eight (x, y) pairs. These pairs are positions that are recorded after fixed intervals when the digit is written. The anomalies are 10 observations from the digits '0', '1', '2', '3', '4', '5', '6', '7', '9' each, leading to 90 anomalies.

6.3.3.3 Letter

This data set consists of features extracted from 3 letters from the English alphabet. The outliers consist of the same features but extracted from the other letters of the alphabet. To make the anomaly detection task more challenging, the contributors added randomly some features to each observation, coming from all letters of the English alphabet.

6.3.3.4 Speech

This data set comes from the domain of speech recognition. Each observation is a so-called 'i-vector representation' of a speech segment. The normal cases come from persons with an American accent, while outliers consist of persons with other accents.

6.3.3.5 Satellite

This data set consists of features extracted from satellite images. These images are used to determine the soil type. In this data set the soil types: 'red soil', 'gray soil', 'damp gray soil' and 'very damp gray soil' are normal instances. Anomalies were sampled from the classes: 'cotton crop' and 'soil with vegetation stubble'.

6.3.3.6 Pen Local

This data set has the same underlying data set as 'Pen Global'. However, instead of focusing on the digit '8', all digits are kept with the exception of the digit '4'. From the latter digit only the first ten observations are included and these form the anomalies.

6.3.3.7 Annthyroid

This data set is derived from the annthyriod data set, also known as the Thyroid disease data set, and comes from the medical domain. It contains features from patients; normal cases represent healthy patients, the outliers are sampled from the patients

that suffer from hypothyroid cancer. The first fifteen features are binary features, the next six features are continuous.

6.3.3.8 Shuttle

This data set is used in the Statlog project and contains features that are connected to the normal and abnormal functioning of radiators in a NASA space shuttle. The original data set is designed for supervised anomaly detection. The version used in this chapter is adjusted by Goldstein and Uchida [43] mainly by reducing the number of outliers.

6.3.3.9 ALOI

The aloi data set originates from a data set provided by the Amsterdam Library of Object Images (ALOI). This library contains images of objects. This particular data set contains a feature vector of length 27 for each image, derived by apply a HSB color histogram. Such a histogram gives the distribution of colors in an image. Each object is photographed many times under different angles and lighting conditions. The 1.508 observations labeled as anomalies correspond to a few objects selected as anomalies.

6.3.3.10 KDD Challenge 1999

This data set comes from a challenge presented at the Knowledge Discovery and Data Mining conference of 1999. The data set contains artificially created observations that represent HTTP traffic in a computer network. The data set is enriched with observations representing observations typically seen in attacks. The current data set has undergone some data preparations to make it more suitable for testing various anomaly detection algorithms, see [43] for details.

6.4 Results

Table 6.2 summarizes the findings of the experiments. The isolation forest algorithm realizes the highest area under the curve on the ‘shuttle’ data. The RBM reaches first place for ‘breast cancer’ and ‘kdd 1999’, although the first position is shared with HBOS (Histogram-Based Outlier Score) for the latter data set. The auc-values for all algorithms in the benchmark can be found in Table 6.3.

Figure 6.4 displays the overall performance of the algorithms on all data sets. The k-NN and kth-NN algorithms perform the best in general. Only on the ‘annthyroid’

	data set	auto-encoder	isolation forest	RBM	mean bench-mark	best bench-mark	best alg. bench-mark
1	breast cancer	0.9091 ± 0.0040	0.9810 ± 0.0014	**0.9858** ±0.00005	0.9067	0.9827 ± 0.0016	HBOS
2	pen global	0.9420 ± 0.0008	0.9304 ±0.0016	0.8282 ±0.0014	0.7836	0.9872 ± 0.0055	k-NN
3	letter	0.7667 ± 0.0042	0.6337 ±0.0052	0.5794 ±0.0026	0.7850	0.9068 ± 0.0078	LoOP
4	speech	0.4716 ± 0.0002	0.4699 ±0.0052	0.4715 ± 0.0002	0.4936	0.5347 ± 0.0343	LoOP
5	satellite	0.9057 ± 0.0017	0.9479 ±0.0018	0.9060 ±0.0018	0.8734	0.9701 ± 0.0007	k-NN
6	pen local	0.8346 ± 0.0039	0.7828 ±0.0064	0.8220 ±0.0036	0.9129	0.9816 ± 0.0024	LOF
7	ann-thyroid	0.5657 ± 0.0010	0.6456 ±0.0058	0.5089 ±0.0026	0.6312	0.9150 ± 0.0123	HBOS
8	shuttle	0.9881 ± 0.0000	**0.9973** ±0.0002	0.9832 ± 0.0004	0.7684	0.9925 ± 0.0039	rPCA
9	aloi	0.5415 ± 0.0004	0.5408 ±0.0003	0.5311 ±0.0006	0.6229	0.7899 ± 0.0093	LoOP
10	kdd 1999	0.9718 ± 0.0001	0.9656 ±0.0017	**0.9990** ±0.00004	0.7926	0.9990 ± 0.0007	HBOS

Table 6.2: Mean Area Under the Curve (AUC) and standard deviation when applying the anomaly detection algorithms with various settings on the benchmark data sets.

data, these algorithms perform below average. The main characteristic of the 'annthyroid' data are its binary features. These binary features have almost no relation with the outlierness of an observation, but do have a large influence on the distance-based k-NN and kth-NN algorithms. Hence we may say that for distance based methods, scaling is important and may give problems when combining binary (or categorical) features with continuous ones.

The Cluster-Based Local Outlier Factor (CBLOF) is clearly the worst algorithm, although variants of this algorithm (uCBLOF and LDCOF) clearly improve performance considerably. The uCBLOF algorithm even reaches the third place in the ranking of algorithms in Figure 6.4. However it performs always worse than the simpler k-NN and kth-NN algorithms, except for data sets with a large U-index (shuttle, kdd 1999). Here the underlying clustering approach may have its advantages.

The Histogram-Based Outlier Score (HBOS) algorithm and the Local Outlier Probability (LoOP) algorithm deserve attention as well, since they are the top performers

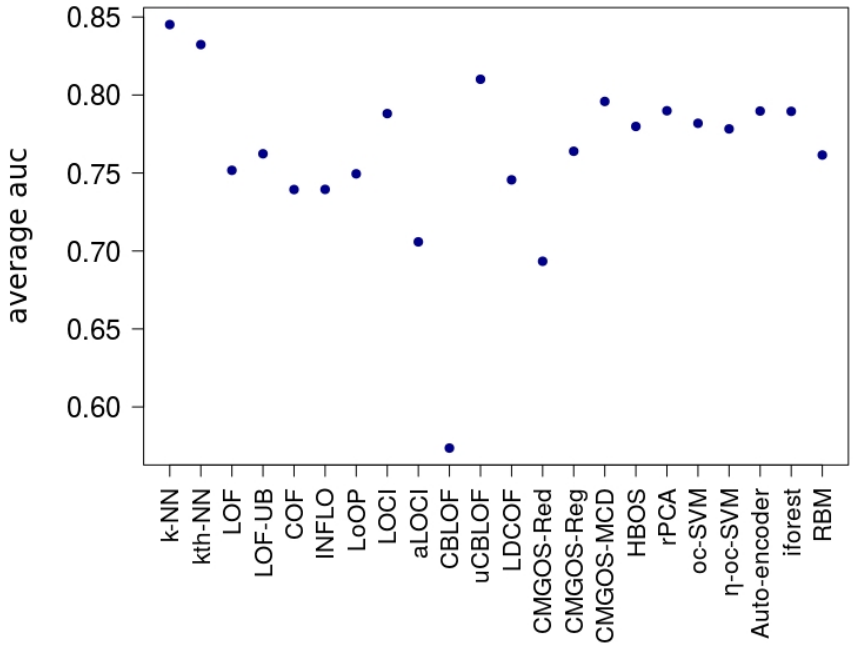

Figure 6.4: Average Area Under the Curve of all algorithms of the benchmark and the three newly added algorithms. The average is taken over all data sets in the benchmark.

for 6 of the 10 data sets among the original 19 algorithms of the benchmark. The simple HBOS algorithm is clearly strong on data sets with a large U-index (an indication for univariate outliers) like breast cancer, shuttle and kdd 1999, while weak on data sets with a small U-index (aloi, letter). It performs also well on annthyroid, where it does not get distracted by the binary features that have almost no correlation with the outliers. In contrast to HBOS, the LoOP algorithm performs well for data sets with a small U-index (letter, pen local, speech, aloi), but badly on data sets with a large U-index (shuttle, kdd 1999).

If we now turn our attention to the newly added algorithms, then we see that all three algorithms are good in finding outliers in data sets with a large U-index (shuttle, kdd 1999, breast cancer, pen global and satellite), often surpassing the currently best algorithm. However the new algorithms perform badly on data sets with a small U-index (letter, speech, aloi). The isolation forest algorithm can best handle the binary features that are present in annthyroid. Because of this property and the fact that isolation forest has the shortest run times and requires almost no tuning of hyperparameters, this algorithm could be labeled as the preferred choice between the three

new algorithms.

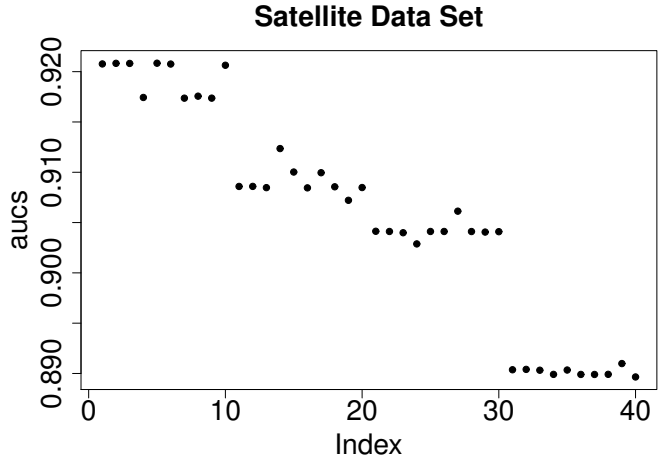

Figure 6.5: Value of the Area Under the Curve for the auto-encoder on the 'Satellite' data.

The time complexity for training one epoch and scoring the data is linear in the number of observations n, the number of features p and the number of hidden nodes h ($\mathcal{O}(nph)$) for standard sparse auto-encoders and RBMs. The time to train and score an isolation forest is linear in the number of observations and independent of the number of features ($\mathcal{O}(n)$). In practice, the run time of the algorithms is to a large extent dependent on the number of epochs needed before reaching convergence during training. With respect to this, isolation forest is the fastest algorithm. From the remaining two, RBMs reached convergence sooner than auto-encoders in our experiments. However this may be caused to a large extent by the use of the BGFS algorithm to optimize the target function for auto-encoders instead of the generally faster stochastic gradient descent method.

After running the experiments for auto-encoders, we have plotted the number of hidden nodes against the auc in order to get more insight in the number of hidden nodes needed in the context of anomaly detection. One of these plots is displayed in Figure 6.5. Observations with index 1 to 10 in this plot are coming from auto-encoders with 5 hidden nodes, index 11 to 20 with 10 hidden nodes, index 21 to 30 with 18 hidden nodes, and finally, index 31 to 40 with 36 hidden nodes. All other hyper-parameters are fixed (except for the random seed that changes for each run). It is clear from the graph that auto-encoders with a small number of hidden nodes are better able to find the outliers (larger auc value). Also for the plots of the other data sets, it is clear that for most data sets 5 or 10 hidden nodes are sufficient.

Algorithm Alg.	breast- cancer	pen- global	pen- local	letter	speech	satel- lite	thy- roid	shuttle	aloi	kdd 1999
k-NN	0.9791	**0.9872**	0.9837	0.8719	0.4966	**0.9701**	0.5956	0.9424	0.6502	0.9747
kth-NN	0.9807	0.9778	0.9757	0.8268	0.4784	0.9681	0.5748	0.9434	0.6177	0.9796
LOF	0.9816	0.8495	**0.9877**	0.8673	0.5038	0.8147	0.647	0.5127	0.7563	0.5964
LOF-UB	0.9805	0.8541	0.9876	0.9019	0.5233	0.8425	0.6663	0.5182	0.7713	0.5774
COF	0.9518	0.8695	0.9513	0.8336	0.5218	0.7491	0.6505	0.5257	0.7857	0.5548
INFLO	0.9642	0.7887	0.9817	0.8632	0.5017	0.8272	0.6542	0.493	0.7684	0.5524
LoOP	0.9725	0.7684	0.9851	**0.9068**	**0.5347**	0.7681	0.6893	0.5049	**0.7899**	0.5749
LOCI	0.9787	0.8877		0.788	0.4979					
aLOCI	0.8105	0.6889	0.8011	0.6208	0.4992	0.8324	0.6174	0.9474	0.5855	0.6552
CBLOF	0.2983	0.319	0.6995	0.6792	0.5021	0.5539	0.5825	0.9037	0.5393	0.6589
uCBLOF	0.9496	0.8721	0.9555	0.8192	0.4692	0.9627	0.5469	0.9716	0.5575	0.9964
LDCOF	0.7645	0.5948	0.9593	0.8107	0.4366	0.9522	0.5703	0.8076	0.5726	0.9873
Red	0.914	0.5693	0.9727	0.7711	0.5077	0.9054	0.4395	0.5425	0.5852	0.7265
Reg	0.8992	0.6994	0.9449	0.8902	0.5081	0.9056	0.6587	0.5679	0.5855	0.9797
MCD	0.9196	0.6265	0.9038	0.7848		0.912	0.8014	0.6903	0.5547	0.9696
HBOS	0.9827	0.7477	0.6798	0.6216	0.4708	0.9135	**0.915**	0.9925	0.4757	**0.999**
rPCA	0.9664	0.9375	0.7841	0.8095	0.5024	0.9461	0.6574	0.9963	0.5621	0.7371
oc-SVM	0.9721	0.9512	0.9543	0.5195	0.465	0.9549	0.5316	0.9862	0.5319	0.9518
η-oc-SVM	0.9581	0.8993	0.9236	0.7298	0.4649	0.943	0.5625	0.9848	0.5221	0.7945
Auto-enc.	0.9091	0.942	0.8346	0.7667	0.4716	0.9057	0.5657	0.9881	0.5415	0.9718
iforest	0.981	0.9304	0.7828	0.6337	0.4699	0.9479	0.6456	**0.9973**	0.5408	0.9656
RBM	**0.9858**	0.8282	0.822	0.5794	0.4715	0.906	0.5089	0.9832	0.5311	**0.999**

Table 6.3: Mean Area Under the Curve for all data sets and algorithms in the benchmark.*Red* stands for CMGOS-Red, *Reg* stands for CMGOS-Reg, *MCD* stands for CMGOS-MCD

6.5 Conclusions and Discussion

Table 6.3 shows the performance of all algorithms on the benchmark of Goldstein and Uchide [43], including the three newly added algorithms. A main observation is that the three algorithms are able to meet or beat the current best algorithm in the benchmark several times: isolation forest is superior on the 'shuttle' data set, while the RBM outperforms all current algorithms on 'breast cancer' and matches the best performance on the 'kdd 1999' data set, see Table 6.2.

Isolation Forest seems to be the preferred choice of the three algorithms that are newly added to the benchmark; the area under the curve on the benchmark is rather similar to the other two algorithms, but it has shorter run times and its hyper-parameters are easy to set. Moreover it handles the binary features in the 'annthyroid' data set well.

Another observation is that there are (at least) two types of data sets with outliers: in the first type of data sets there is at least one feature that has a correlation with the label indicating an outlier. Anomaly detection algorithms that perform well on these data sets are: HBOS, rPCA, oc-SVM, η-oc-SVM, auto-encoder, isolation forest, RBM, and uCBLOF. We see that all three new algorithms fall in this category. The second type of data sets lack such a univariate feature. On these data sets LOF-like (Local Outlier Factor) algorithms perform well: LOF, LOF-UB, COF, INFLO, and LoOP. The U-index, as introduced by equation (6.13), helps to classify the data sets in the benchmark in these two groups, see Table 6.1.

Further, it is noteworthy that simplicity seems to go hand in hand with power at several places. This is evident in Figure 6.4, where it becomes clear that the simple algorithms of k-NN and kth-NN perform the best when considering the average performance on all data sets. Also, Table 6.2 shows that the relatively simple HBOS algorithm belongs to the top performers for data sets with a large U-index. Finally, we note that simplicity in the number of nodes (i.e. a small number) for auto-encoders and RBMs does not decrease performance, see for instance Figure 6.5.

When reflecting on the results, then some reservations are in order. First, the benchmark contains a limited number of data sets (ten), and a disproportional large number of these data sets are features extracted from images ('breast cancer', 'pen global', 'pen local', 'letter', 'satellite', 'aloi'). Moreover all data sets are tabular data and contain no categorical features. Second, in this chapter only the most common implementations of the three newly added algorithms are tested. Each algorithm knows extensions that are worth further investigation in the future: the isolation forest algorithm has recently been extended by allowing splits in the feature space that are not parallel to coordinate axes, see the paper of Hariri et al. [48]. Auto-encoders can be extended by allowing more hidden layers, while also interesting variants exist that

are mainly developed for large data sets (variational auto-encoders, GAN-networks [103]). RBMs can be stacked, leading to deep belief networks. Also a combination of auto-encoders and RBMs exists, see the paper of Hinton and Salakhutdinov [52]. Here RBMs initialize the weights in deep auto-encoders, making it feasible to train these networks with gradient descent.

For further research we also want to mention the 'speech' data set. None of the current algorithms performs well on this data set due to the combination of many features (400) and a low U-index.