Journal of Machine Learning Research 21 (2020) 1-22 Submitted 6/19; Revised 12/19; Published 3/20

$(1+\varepsilon)$-class Classification: an Anomaly Detection Method for Highly Imbalanced or Incomplete Data Sets

Maxim Borisyak MBORISYAK@HSE.RU
Artem Ryzhikov ARYZHIKOV@HSE.RU
Andrey Ustyuzhanin AUSTYUZHANIN@HSE.RU
Denis Derkach DDERKACH@HSE.RU
Fedor Ratnikov FRATNIKOV@HSE.RU
Olga Mineeva OMINEEVA@ETHZ.CH
Laboratory of Methods for Big Data Analysis
National Research University Higher School of Economics
20 Myasnitskaya ulitsa, Moscow 101000 Russia

Editor: Miguel Carreira-Perpinan

Abstract

Anomaly detection is not an easy problem since distribution of anomalous samples is unknown a priori. We explore a novel method that gives a trade-off possibility between one-class and two-class approaches, and leads to a better performance on anomaly detection problems with small or non-representative anomalous samples. The method is evaluated using several data sets and compared to a set of conventional one-class and two-class approaches.

Keywords: Anomaly Detection, Imbalanced Data Sets, Neural Networks, One-class Classification, Regularization

1. Introduction

Monitoring of complex systems and processes often goes hand in hand with anomaly detection. Anomaly here means a representation of abnormal system behavior. Information on normal system behavior is often available in abundance, compared to samples of abnormal behavior. In some cases the anomalies are rare, or distribution of anomalies is highly skewed. So, given the high variability of anomalies, it leads to the fact that some types of anomalies are missing in the training data set. In other cases, when anomalous examples are obtained by means other than sampling target system, or when the distribution of anomalies evolve over time, some types of anomalies might even be unknown in principle. A good realistic data set with anomalous behavior is provided by KDD-99 Cup (KDD, 1999), with certain families of cyber-attacks present only in the test sample.

Conventional approaches for anomaly detection often involve one-class classification methods (Chalapathy et al., 2018; Tax and Duin, 2001; Ruff et al., 2018; Liu et al., 2008; Scholkopf and Smola, 2001), which yield a soft boundary between the normal class region, and the rest of the feature space. Usually such methods are referred to as unsupervised, since those do not take into account labels of available data. As this piece of information might be important, those one-class methods potentially lead to the performance degra-

dation for the cases with significant overlap between normal and abnormal samples in the feature space.

There is a rich profusion of two-class supervised classification methods that account for both class labels, leading to better results in the presence of labeled abnormal samples. However, those methods lack any guarantees for predictions outside of the regions of the feature space presented in the training data. It becomes especially problematic for incomplete anomalous samples, as a classifier might consistently make false-positive predictions for unseen anomalies.

Contribution In this study, we develop a method that is aimed at combining the best of the two, one-class and two-class approaches, which we refer to as $(1 + \varepsilon)$-class classification (*'one plus epsilon'* or *OPE* for short). In order to achieve that, we derive two one-class objectives and combine them with the binary cross-entropy loss. We compare these objectives with respect to computational effectiveness, and demonstrate performance on several data sets that are either collected for anomaly detection tasks (KDD, 1999), or artificially under-sampled to emulate these conditions (Baldi et al., 2014; LeCun et al., 1998; Krizhevsky and Hinton, 2009; Lake et al., 2015).

Notation We assume that an N-dimensional feature space $\mathcal{X}$ ($\mathbb{R}^N$), contains samples of two classes: normal (positive) $\mathcal{C}^+$ and abnormal (negative) $\mathcal{C}^-$. We are interested in identifying instances of the single class $\mathcal{C}^+$. There are two principal approaches: one-class (unitary classification) and two-class (binary classification). The former might rely on estimation of the likelihood of the positive class $P(x \mid \mathcal{C}^+)$, so then one can apply a threshold to make a final decision. We will refer to any solution of the form $s(P(x \mid \mathcal{C}^+))$, where $s : \mathbb{R} \to \mathbb{R}$ is a monotone function, as a *unitary classification solution*. The latter relies on estimation of the posterior conditional distribution $P(\mathcal{C}^+ \mid x)$, that is usually approximated through minimization of the cross-entropy loss function:

$$\mathcal{L}_2(f) = P(\mathcal{C}^+) \mathop{\mathbb{E}}_{x\sim\mathcal{C}^+} \log f(x) + P(\mathcal{C}^-) \mathop{\mathbb{E}}_{x\sim\mathcal{C}^-} \log\left(1 - f(x)\right); \tag{1}$$

where $\mathbb{E}_{x\sim\mathcal{C}}\, h(x)$ denotes conditional expectation $\mathbb{E}_x\left[h(x) \mid \mathcal{C}\right]$, and $f : \mathcal{X} \to [0, 1]$—classifier's decision function.

Optimal binary decision function f^* that minimizes $\mathcal{L}_2(f)$, can be expressed with the help of Bayes' rule as

$$f^*(x) = P(\mathcal{C}^+ \mid x) = \frac{P(x \mid \mathcal{C}^+)P(\mathcal{C}^+)}{P(x \mid \mathcal{C}^+)P(\mathcal{C}^+) + P(x \mid \mathcal{C}^-)P(\mathcal{C}^-)}; \tag{2}$$

where $P(\mathcal{C})$ is class prior probability, $P(\mathcal{C} \mid x)$—posterior conditional distribution and $P(x \mid \mathcal{C})$—likelihood for the given class $\mathcal{C}$.

2. One Plus Epsilon Method

Let's consider a simple case: $\mathcal{C}^-$ is a uniform distribution $U[\Omega]$, with the support Ω covering that of $P(x \mid \mathcal{C}^+)$. If we put this $\mathcal{C}^-$ into Equation (1) (assuming equal class priors), we get

$$\begin{aligned}
\mathcal{L}_1(f) &= -\frac{1}{2}\left[\mathop{\mathbb{E}}_{x\sim\mathcal{C}^+} \log f(x) + \mathop{\mathbb{E}}_{x\sim U[\Omega]} \log(1 - f(x))\right]; \\
f_1^*(x) &= \arg\min_f \mathcal{L}_1(f) = \frac{P(x \mid \mathcal{C}^+)}{P(x \mid \mathcal{C}^+) + C};
\end{aligned}$$

where C—probability density of distribution $U[\Omega]$. Note, that $f_1^*(x)$ is a unitary classification solution, therefore, *solution to a classification problem between a given class and a uniformly distributed one, yields a unitary classification solution.*

2.1. Adding Known Negative Samples

Let's take into account known anomalous samples. We propose the following loss function—linear combination of one-class classification loss $\mathcal{L}_1$ and cross-entropy loss $\mathcal{L}_2$:

$$\begin{aligned} \mathcal{L}_{1+\varepsilon}(f) &= \frac{1}{2}\left(L^+(f) + \gamma\, L^-(f) + (1-\varepsilon)\, L^0(f)\right); \\ L^+(f) &= -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^+} \log f(x); \\ L^-(f) &= -\mathop{\mathbb{E}}_{x\sim\mathcal{C}^-} \log(1-f(x)); \\ L^0(f) &= -\mathop{\mathbb{E}}_{x\sim U} \log(1-f(x)); \end{aligned} \tag{3}$$

where γ compensates for the difference in classes prior probabilities. Ideally, it should be set to $P(\mathcal{C}^-)/P(\mathcal{C}^+)$, so that the first two terms match the cross-entropy loss. ε is a hyper-parameter, that allows to choose the trade-off between unitary and binary classification solutions. We call the loss $\mathcal{L}_{1+\varepsilon}(f)$ *OPE loss*. It leads to the following solution:

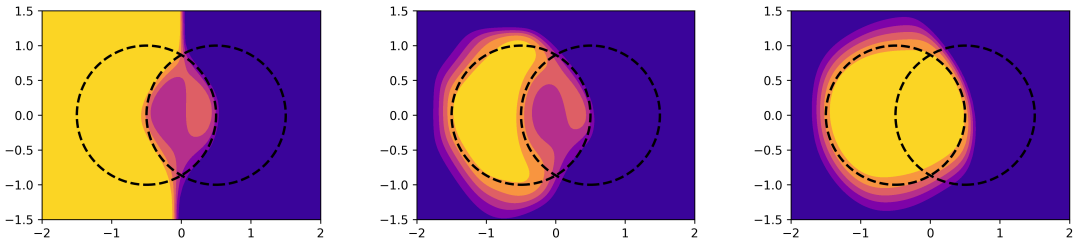
$$f^*_{1+\varepsilon}(x) = \frac{P(x \mid \mathcal{C}^+)}{P(x \mid \mathcal{C}^+) + (1-\varepsilon)\, C + \gamma\, P(x \mid \mathcal{C}^-)}.$$

An important observation can be made—for large capacity models, even ε close to 1 leads to a significantly different solution in comparison to the two-class classification one (Equation 2). This effect can be observed in Figure 1, which shows predictions of networks trained to minimize $\mathcal{L}_2$, $\mathcal{L}_{1+\varepsilon}$ and $\mathcal{L}_1$ loss functions on a synthetic data set. Intuitively, for a sufficiently large network, there are multiple solutions that achieve nearly optimal cross-entropy loss, including the ones that make positive predictions outside supp $\mathcal{C}^+$, similar to the solution in Figure 1a. Since cross-entropy loss function does not depend on predictions outside supp $\mathcal{C}^+ \cup$ supp $\mathcal{C}^-$, a randomly initialized network is likely to converge to one of such solutions. $\mathcal{L}_{1+\varepsilon}$ loss function explicitly penalizes positive predictions everywhere, thus, making solutions with unitary classification properties more preferable than others.

One might consider the term L_0 as a regularization term, that biases solution $f^*_{1+\varepsilon}(x)$ towards 0 everywhere, but this effect is especially pronounced in points with $P(x \mid \mathcal{C}^+) \approx 0$. One distinguishing feature of the L_0 regularization term is that it acts directly on predictions, rather then on parameters[1], which makes it applicable to any classifier model.

Estimating L^0 term in Equation (3) for low-dimensional feature-space is straightforward—if supp $P(x \mid \mathcal{C}^+)$ can be bounded by a simple set Ω (e.g., a box), then L_0 can be estimated by directly sampling from $U[\Omega]$. We refer to this class of OPE algorithms as *brute-force OPE*.

1. Technically, regularization in one-class SVM objective (Schölkopf et al., 2000) and similar methods can also be considered to act directly on predictions since these are linear models.



(a) two-class classification (b) OPE classification (c) one-class classification.

Figure 1: Demonstration of the main idea behind OPE loss. Samples are uniformly distributed within areas bounded by the circles: the left one as positive class, the right one as negative. One-class solution was obtained by setting $\gamma = 1$, and $\varepsilon = 0$. Training samples are not shown for visual clarity.

2.2. Energy-Based Regularization

For high-dimensional feature space, however, sampling directly from $U[\Omega]$ might be problematic, due to a potentially high variance of the gradients produced by the regularization term. One possible strategy of reducing variance of L^0 gradient estimates, is to sample from another distribution Q:

$$L^0(f) = \mathop{\mathbb{E}}_{x\sim U} \log(1 - f(x)) = \mathop{\mathbb{E}}_{x\sim Q} \frac{C}{Q(x)} \log(1 - f(x)); \tag{4}$$

Jin et al. (2017) employs this method and uses distribution $Q = P_f$ induced by the model f at the previous training epoch:

$$P_f(x) = \frac{1}{Z} \frac{\mathbb{I}[x \in \Omega] \cdot f(x)}{1 - f(x)};$$

where: $Z = \int_\Omega \frac{f(x)}{1-f(x)} dx$—normalization term, $\mathbb{I}$—indicator function. Hence, L^0 can be written as:

$$L^0(f) = Z \cdot \mathop{\mathbb{E}}_{x\sim P_f} C \cdot \frac{1 - f(x)}{f(x)} \log(1 - f(x)).$$

Sampling from P_f is computationally expensive, and various methods can be used, e.g., Hamiltonian Monte-Carlo (Duane et al., 1987). However, this transformation merely transfers the computationally heavy integration part from uniform sampling to estimation of the normalization term Z. In order to avoid recomputing Z on each epoch, a two-stage training procedure is proposed by Jin et al. (2017) and Tu (2007):

1. freeze sampling distribution P_f, estimate Z;
2. using this frozen distribution perform a number of stochastic gradient descent steps.

Note, that as long as a regularization term shifts the decision function towards 0 outside of $\operatorname{supp} P(x \mid \mathcal{C}^+)$, and has a small impact within, it suffices for the purposes of anomaly

detection. With that idea in mind, we propose the following approximation of L^0 regularization term to avoid uniform sampling and integration.

Let's introduce $g(x) = \sigma^{-1}(f)$, where $\sigma(\chi) = 1/\left[1+\exp(-\chi)\right]$—sigmoid function:

$$\begin{aligned} P_g(x) &= \frac{1}{Z}\frac{f(x)}{1-f(x)} = \frac{1}{Z}\exp(g(x)); \\ \text{where } Z &= \int_\Omega \exp(g(x))dx. \end{aligned} \tag{5}$$

Note, that $g(x)$ in Equation (5) matches the definition of (negative) energy $E(x)$ used in energy-based generative models (Bengio et al., 2009): $P(x) \propto \exp(-E(x))$.

In case of $Z \gg C$, using Jensen inequality, we can approximate upper bound of L^0 as follows:

$$L^0 = -\mathop{\mathbb{E}}_{x\sim U}\log(1-f(x)) = \mathop{\mathbb{E}}_{x\sim U}\log(1+\exp(g(x))) \le \log\left[1+\mathop{\mathbb{E}}_{x\sim U}\exp(g(x))\right] = \log\left(1+\frac{Z}{C}\right) \approx \log Z - \log C;$$

which leads to the following one-class loss function:

$$\begin{aligned} \mathcal{L}_1^E(g) &= \frac{1}{2}\left[\mathop{\mathbb{E}}_{x\sim\mathcal{C}^+}\log\left(1+\exp(-g(x))\right) + (1-\varepsilon)L^E(g)\right]; \\ \text{where } L^E(g) &= \log Z = \int_\Omega \exp(g(x))dx; \end{aligned} \tag{6}$$

then the corresponding energy OPE (EOPE) loss function is

$$\mathcal{L}_{1+\varepsilon}^E(f) = \frac{1}{2}\left(L^+(f) + \gamma L^-(f) + (1-\varepsilon)L^E(\sigma^{-1}(f))\right). \tag{7}$$

Gradients of L_E^0 can be easily estimated (see, for example, Bengio et al., 2009):

$$\nabla L^E(g) = \nabla\log Z = \frac{1}{Z}\int_\Omega \exp(g(x))\nabla g(x) = \mathop{\mathbb{E}}_{x\sim P_g}\nabla g(x). \tag{8}$$

Note, that Equation (8) essentially describes the negative phase of contrastive divergence algorithm for energy-based models. Similar relations between the cross-entropy loss and contrastive divergence have also been mentioned by Kim and Bengio (2016).

As discussed above, the main goal of L^0 regularization term is to enforce one-class properties, namely, make the solution to be a monotone transformation of $P(x \mid \mathcal{C}^+)$. The following theorem shows that, despite being just an approximation of L^0 regularization, $\mathcal{L}_1^E(g)$ *loss always leads to a one-class solution.*

Theorem 1 *Let $(\mathcal{X}, \|\cdot\|)$ be a Banach space, $P(x)$—a continuous probability density function such that $\Omega = \operatorname{supp} P$ is an open set in $\mathcal{X}$. If continuous function $g^* : \Omega \to \mathbb{R}$ minimizes $\mathcal{L}_1^E$ (defined by Equation 6) with $P(x \mid \mathcal{C}^+) = P(x)$, then there exists a strictly*

Algorithm 1: Brute-force OPE

Input: normal data, anomalous data—samples from $\mathcal{C}^+$, $\mathcal{C}^-$, the latter might be absent; f_θ—a classifier with parameters θ.

Hyper-parameters: γ—ratio of class priors; ε—controls strength of regularization.

while *not converged* **do**

sample normal data $\{x_i^+ \sim \text{normal data}\}_{i=1}^m$;

sample known anomalies $\{x_i^- \sim \text{anomalous data}\}_{i=1}^m$;

sample negative examples $\{x_i^0 \sim U[\Omega]\}_{i=1}^m$;

$\nabla L^+ \leftarrow -\sum_i \nabla_\theta \log f_\theta(x_i^+)$;

$\nabla L^- \leftarrow -\sum_i \nabla_\theta \log(1 - f_\theta(x_i^-))$;

$\nabla L^0 \leftarrow -\sum_i \nabla_\theta \log(1 - f_\theta(x_i^0))$;

$\theta \leftarrow \text{Adam}\left(\nabla L^+ + \gamma \nabla L^- + (1-\varepsilon)\nabla L^0\right)$

end

increasing function $s : \mathbb{R} \to \mathbb{R}$*, such that* $g^*(x) = s(P(x))$*. Moreover,* $\lim_{y\to 0} s(y) = -\infty$ *(if* $\inf_\Omega P = 0$*).*

Intuitively, it is clear, that if the dependency between $g(x)$ and $P(x)$ is violated in some regions, energy can be exchanged between these regions with a total reduction in the loss. A similar argument can be made for the property: $\lim_{y\to 0} s(y) > -\infty$—energy of low-density regions can be transferred to a high-density region, leading to an improved solution. A more formal proof can be found in Appendix B.

3. Implementation Details

While OPE and EOPE losses are independent of any particular choice of model f, we consider only neural networks. We optimize all neural networks with a stochastic gradient method (namely, Adam algorithm by Kingma and Ba, 2014). Algorithms 1 and 2 outline proposed methods.

Estimation of $L^E(f)$ is tightly linked to the negative phase of energy-based generative models. A traditional approach for sampling from P_f is to employ Monte-Carlo (MC) methods, in this work we use Hamiltonian Monte-Carlo (HMC). Additionally, in our experiments we use persistent MC chains following Tieleman (2008). Nevertheless, usage of MC leads to a significant slow down of the training procedure, as in general, multiple passes through the network are required for generating negative samples.

Note, that for values of ε close to 1, both L^0 and L^E have a significant impact only in the regions with low probability density $P(x \mid \mathcal{C}^+)$. This suggests that solutions of Equations (3) and (7) are relatively robust to improper sampling procedures, and one might achieve a faster training without sacrificing much of quality, by employing fast approximate MC procedures. In our experiments we observed that the following highly degenerate

Algorithm 2: Energy OPE

Input: normal data, anomalous data—samples from $\mathcal{C}^+$, $\mathcal{C}^-$, the latter might be absent; g_θ—a classifier with parameters θ.

Hyper-parameters: γ—ratio of class priors; ε—controls strength of regularization; MCMC—Monte-Carlo sampling procedure.

while *not converged* **do**

sample normal data $\{x_i^+ \sim \text{normal data}\}_{i=1}^m$;
sample known anomalies $\{x_i^- \sim \text{anomalous data}\}_{i=1}^m$;
sample negative examples $\{x_i^0 \sim \text{MCMC}\,[x \mapsto \exp(g(x))]\}_{i=1}^m$;
$\nabla L^+ \leftarrow \sum_i \nabla_\theta \log(1+\exp(-g_\theta(x_i^+)))$;
$\nabla L^- \leftarrow \sum_i \nabla_\theta \log(1+\exp(g_\theta(x_i^-)))$;
$\nabla L^E \leftarrow \sum_i \nabla_\theta g_\theta(x_i^0)$;
$\theta \leftarrow \text{Adam}\left(\nabla L^+ + \gamma \nabla L^- + (1-\varepsilon)\nabla L^E\right)$

end

instance of HMC is performing well:

$$x_{t+1} = x_t + \eta\left[\frac{\nabla g(x)}{\sqrt{m_{t+1}}} + \lambda\xi_t\right]; \tag{9}$$

$$m_{t+1} = \rho m_t + (1-\rho)\big(\nabla g(x) \odot \nabla g(x)\big); \tag{10}$$

where: $\odot$ denotes Hadamard product, ξ_t is distributed normally with zero mean and unit covariance matrix, $\lambda > 0$ controls the impact of the random noise, $\eta > 0$—step size, $0 < \rho < 1$—coefficient for the moving average m_t. We refer to the methods utilizing such sampling as RMSProp-EOPE, since the procedure resembles RMSProp optimization algorithm (Tieleman and Hinton, 2012).

A completely different approach to negative phase sampling is described by Kim and Bengio (2016). The authors suggest using a separate network (generator) to produce samples from the target distribution. We also implement this sampling procedure and refer to the methods employing it as Deep EOPE.

In our experiments, we observe that methods based on EOPE loss function, quickly lead to steep functions which heavily interfere with the sampling procedures. Following Tieleman and Hinton (2009), we add a small l_2 regularization term for predictions in pseudo-negative points:

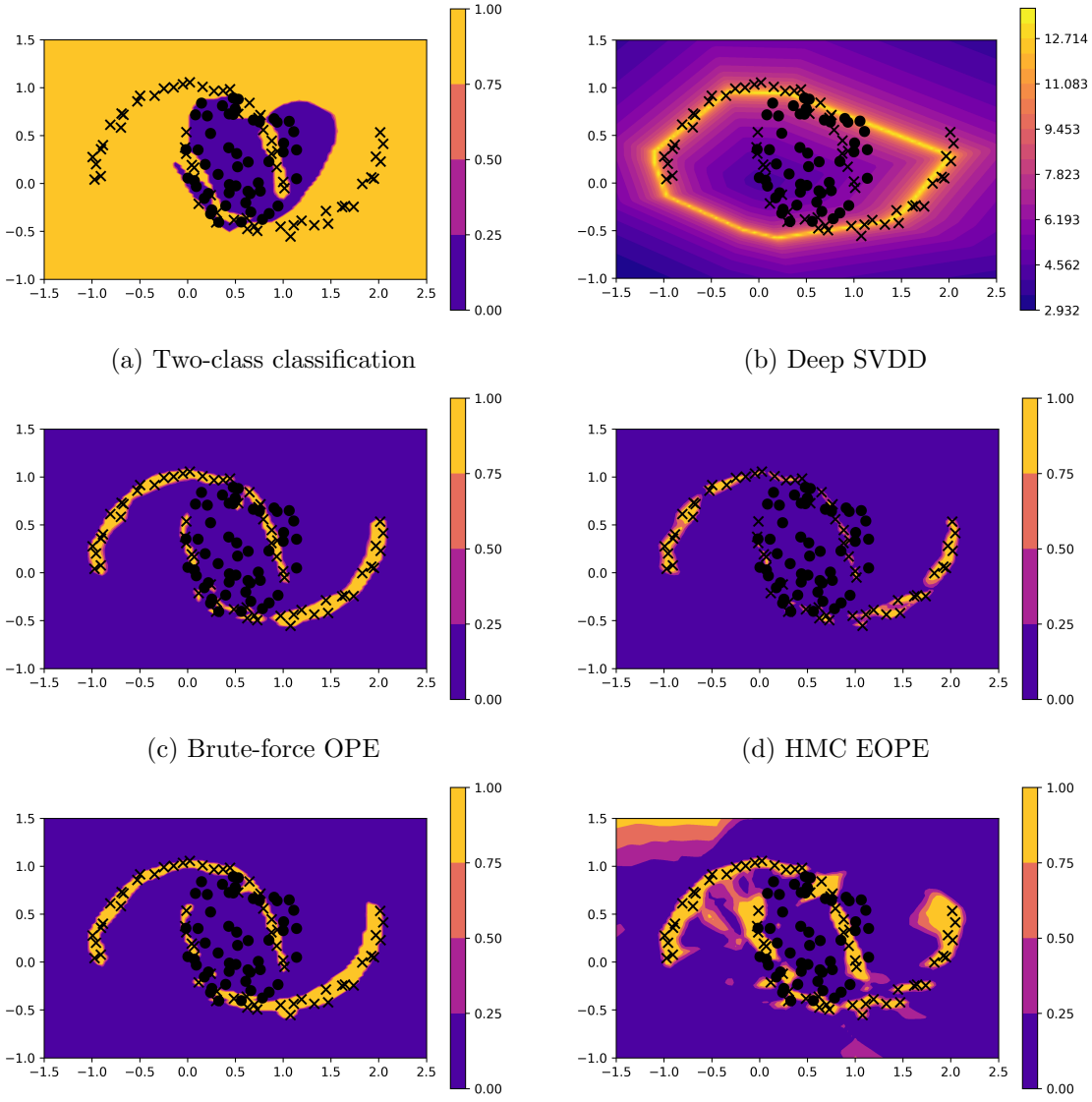
$$\tilde{L}^E(g) = \int_\Omega \exp(g(x))dx + c \mathop{\mathbb{E}}_{x\sim P_f} \|g(x)\|^2;$$

where c is a small constant ($c = 10^{-3}$ in our experiments).

Figures 2 and 3 demonstrate results of proposed methods on a toy data set.

4. Relation to Other Methods

The idea to perform one-class classification (and generative task) as 'one against everything', appears in many studies. Tax and Duin (2001) propose constructing a hyper-sphere around positive samples, effectively separating it from the rest of the space; Ruff et al. (2018) and



(a) Two-class classification

(b) Deep SVDD

(c) Brute-force OPE

(d) HMC EOPE

(e) RMSProp EOPE

(f) Deep EOPE

Figure 2: Comparison of different methods on a toy example: positive examples (marked as 'x') are sampled from the Moons data set, negative examples (marked by black circles) are sampled uniformly from a circle of radius $\frac{1}{2}$. For visual consistency negative logarithm of Deep SVDD output is displayed.

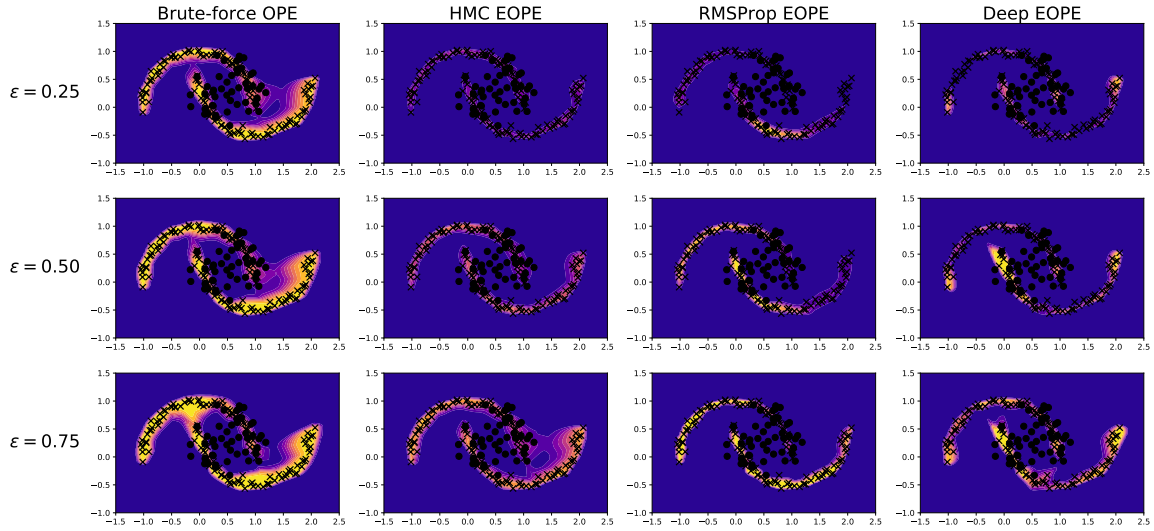


Figure 3: Comparison of OPE and EOPE losses with varying ε, and, for illustration purposes, $\gamma = 1 - \varepsilon$. For $\varepsilon < 1$, all losses lead to similar solutions. It appears that EOPE loss tends to overpenalize large predictions in contrast to OPE loss.

Chalapathy et al. (2018) extend this idea on deep neural networks. Ruff et al. (2018) rely on weight regularization, which acts in a similar manner to EOPE by limiting the area with high model output. OPE and EOPE methods depend only on the model's output, which allows OPE and EOPE methods to avoid limiting number of layers (for example, Chalapathy et al., 2018), and does not restrict choice of network architecture (Ruff et al., 2018).

Tu (2007) and Jin et al. (2017) developed a method similar in its nature to OPE, in fact, it is easy to see, that L_0 term as it appears in Equation (4), corresponds to the loss function by Jin et al. (2017). In this work we demonstrate that this loss is equivalent to the cross-entropy loss between a given class and a uniform distribution covering its support. EOPE loss alleviates computational expenses associated with the estimation of the normalization term and the RMSProp-like sampling procedure further accelerates training by reducing computational costs of sampling.

Learning from positive and unlabeled data (PU learning, Bekker and Davis, 2018) is a closely related field. The problem statement of PU learning is somewhat similar to that of OPE—binary classification with labeled positive samples and an unlabeled mixture of negative and positive samples. However, there are substantial differences between OPE and PU learning settings: in this study, we focus primarily on the case of a non-representative anomalous sample rather than on incomplete label information; nevertheless, some analogies might be drawn. Most notably, some PU learning approaches consider unlabeled part of the data set as negative class, which resembles 'one against everything' approach (for example, Elkan and Noto, 2008; Northcutt et al., 2017).

Li and Liu (2005) consider a PU learning problem when the negative class might be different at test time, i.e., training negative sample is non-representative. Authors suggest

enriching unlabeled sample with a large set of 'irrelevant' examples which is unlikely to contain any significant number of positive samples and then introduce a modified EM algorithm for training a classifier. This method can also be viewed as 'one against everything' approach employed by brute-force OPE.

Basile et al. (2017) also consider PU learning settings and suggest training a generative model on positive samples and then labeling samples that fall into low-density regions as reliably negative. This approach resembles EOPE, however, the most notable difference is that EOPE employs contrastive divergence, while the former method relies on the assumption that low-density regions are associated with the negative class.

5. Experiments

We evaluate proposed methods on the following data sets: MNIST (LeCun et al., 1998), CIFAR (Krizhevsky and Hinton, 2009), KDD-99 (KDD, 1999), Omniglot (Lake et al., 2015), SUSY and HIGGS (Baldi et al., 2014). In order to reflect the assumptions behind our approach we derive multiple tasks from each data set by varying size of the anomalous subset.

As the proposed methods target problems intermediate between one-class and two-class problems, we compare our approaches against the following algorithms:

- conventional two-class classification with the cross-entropy loss;
- a semi-supervised method: dimensionality reduction by a deep AutoEncoder followed by a classifier with the cross-entropy loss;
- one-class methods: Deep SVDD (Ruff et al., 2018) and Robust AutoEncoder (Zhou and Paffenroth, 2017).

Since not all of the evaluated algorithms allow for a probabilistic interpretation, ROC AUC metric is reported. As performance of certain algorithms (especially, two-class classification) varies significantly depending on the choice of negative class, we run each experiment multiple times, and report average and standard deviation of the metrics. The results are reported in Tables 4–9. Detailed description of the experimental setup can be found in Appendix A.

In these tables, columns represent tasks with varying numbers of negative samples presented in the training set: numbers in the header indicate either number of classes that form a negative class (in case of MNIST, CIFAR, Omniglot and KDD data sets), or number of negative samples used (HIGGS and SUSY); 'one-class' denotes absence of known anomalous samples. As one-class algorithms do not take into account negative samples, results of these are repeated for the tasks with known anomalies.

In our experiments, we make several observations. Firstly, proposed methods generally outperform baseline methods, especially on the problems with a significant overlap between classes (SUSY, HIGGS and, possibly, CIFAR), and consistently show comparable performance on test problems. Secondly, we observe increasing performance as more negative samples are included in the training set, while being consistently above or similar to that of conventional two-class classification. Lastly, to our surprise, brute-force OPE performs

	one class	100	1000	10000	1000000
Robust AE	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002	0.530 ± 0.002
Deep SVDD	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006	0.497 ± 0.006
cross-entropy	-	0.496 ± 0.017	0.529 ± 0.007	0.566 ± 0.006	0.858 ± 0.002
semi-supervised	-	0.498 ± 0.003	0.522 ± 0.003	0.603 ± 0.002	0.745 ± 0.005
brute-force OPE	0.499 ± 0.009	0.500 ± 0.009	0.520 ± 0.003	0.572 ± 0.005	0.859 ± 0.001
HMC EOPE	0.491 ± 0.000	0.523 ± 0.005	$\mathbf{0.567} \pm 0.008$	$\mathbf{0.648} \pm 0.005$	0.848 ± 0.001
RMSProp EOPE	0.498 ± 0.002	0.494 ± 0.008	0.531 ± 0.008	0.593 ± 0.011	$\mathbf{0.861} \pm 0.000$
Deep EOPE	$\mathbf{0.531} \pm 0.000$	$\mathbf{0.537} \pm 0.011$	0.560 ± 0.008	0.628 ± 0.005	0.860 ± 0.001

Figure 4: Results on HIGGS data set. The first row indicates numbers of negative samples used in training.

	one class	100	1000	10000	1000000
Robust AE	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012	0.394 ± 0.012
Deep SVDD	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022	0.541 ± 0.022
cross-entropy	-	0.658 ± 0.033	0.736 ± 0.021	0.757 ± 0.036	0.871 ± 0.006
semi-supervised	-	0.715 ± 0.020	0.766 ± 0.009	$\mathbf{0.847} \pm 0.002$	0.876 ± 0.000
brute-force OPE	$\mathbf{0.648} \pm 0.035$	0.678 ± 0.025	0.729 ± 0.029	0.757 ± 0.036	0.871 ± 0.006
HMC EOPE	0.472 ± 0.000	$\mathbf{0.738} \pm 0.019$	$\mathbf{0.770} \pm 0.012$	0.816 ± 0.006	0.877 ± 0.000
RMSProp EOPE	0.443 ± 0.038	0.714 ± 0.019	0.760 ± 0.016	0.807 ± 0.004	0.877 ± 0.000
Deep EOPE	0.468 ± 0.118	0.670 ± 0.054	0.746 ± 0.024	0.813 ± 0.003	$\mathbf{0.878} \pm 0.000$

Figure 5: Results on SUSY data set. The first row indicates numbers of negative samples used in training.

	one class	1	2	4	8
Robust AE	$\mathbf{0.972} \pm 0.006$	$\mathbf{0.972} \pm 0.006$	$\mathbf{0.972} \pm 0.006$	$\mathbf{0.972} \pm 0.006$	$\mathbf{0.972} \pm 0.006$
Deep SVDD	0.939 ± 0.014	0.939 ± 0.014	0.939 ± 0.014	0.939 ± 0.014	0.939 ± 0.014
cross-entropy	-	0.571 ± 0.213	0.300 ± 0.182	0.687 ± 0.268	0.619 ± 0.257
semi-supervised	-	0.315 ± 0.258	0.469 ± 0.286	0.758 ± 0.171	0.865 ± 0.087
brute-force OPE	0.398 ± 0.108	0.667 ± 0.175	0.394 ± 0.261	0.737 ± 0.187	0.541 ± 0.257
HMC EOPE	0.786 ± 0.200	0.885 ± 0.152	0.919 ± 0.055	0.863 ± 0.094	0.958 ± 0.023
RMSProp EOPE	0.765 ± 0.216	0.824 ± 0.237	0.770 ± 0.213	0.941 ± 0.048	0.960 ± 0.021
Deep EOPE	0.602 ± 0.279	0.767 ± 0.245	0.548 ± 0.279	0.763 ± 0.217	0.786 ± 0.267

Figure 6: Results on KDD-99 data set. The first row indicates numbers of original classes selected as negative class, at most 1000 examples are sampled from each original class.

	one class	1	2	4
Robust AE	$\mathbf{0.978} \pm 0.017$	$\mathbf{0.978} \pm 0.017$	0.978 ± 0.017	0.978 ± 0.017
Deep SVDD	0.641 ± 0.086	0.641 ± 0.086	0.641 ± 0.086	0.641 ± 0.086
cross-entropy	-	0.879 ± 0.108	0.957 ± 0.050	0.987 ± 0.014
semi-supervised	-	0.934 ± 0.035	0.964 ± 0.032	0.984 ± 0.012
brute-force OPE	0.786 ± 0.112	0.915 ± 0.096	0.968 ± 0.041	0.986 ± 0.015
HMC EOPE	0.694 ± 0.167	0.933 ± 0.060	0.974 ± 0.023	0.989 ± 0.011
RMSProp EOPE	0.720 ± 0.186	0.933 ± 0.062	0.977 ± 0.023	0.990 ± 0.009
Deep EOPE	0.793 ± 0.129	0.942 ± 0.048	$\mathbf{0.979} \pm 0.016$	$\mathbf{0.991} \pm 0.007$

Figure 7: Results on MNIST data set. The first row indicates numbers of original classes selected as negative class, 10 images are sampled from each original class.

	one class	1	2	4
Robust AE	$\mathbf{0.585} \pm 0.126$	0.585 ± 0.126	0.585 ± 0.126	0.585 ± 0.126
Deep SVDD	0.546 ± 0.058	0.546 ± 0.058	0.546 ± 0.058	0.546 ± 0.058
cross-entropy	-	0.659 ± 0.093	0.708 ± 0.086	0.748 ± 0.082
semi-supervised	-	0.587 ± 0.109	0.634 ± 0.109	0.671 ± 0.093
brute-force OPE	0.549 ± 0.098	$\mathbf{0.688} \pm 0.087$	$\mathbf{0.719} \pm 0.079$	$\mathbf{0.757} \pm 0.073$
HMC EOPE	0.547 ± 0.116	0.678 ± 0.091	0.709 ± 0.084	0.739 ± 0.074
RMSProp EOPE	0.565 ± 0.111	0.678 ± 0.081	0.715 ± 0.083	0.746 ± 0.069
Deep EOPE	0.564 ± 0.094	0.674 ± 0.100	0.690 ± 0.092	0.719 ± 0.099

Figure 8: Results on CIFAR-10 data set. The first row indicates numbers of original classes selected as negative class, 10 images are sampled from each original class.

	one class	1	2	4
Robust AE	$\mathbf{0.771} \pm 0.221$	0.771 ± 0.221	0.771 ± 0.221	0.771 ± 0.221
Deep SVDD	0.640 ± 0.153	0.640 ± 0.153	0.640 ± 0.153	0.640 ± 0.153
cross-entropy	-	0.799 ± 0.162	$\mathbf{0.862} \pm 0.115$	0.855 ± 0.125
semi-supervised	-	0.737 ± 0.134	0.821 ± 0.104	0.805 ± 0.121
brute-force OPE	0.591 ± 0.161	0.724 ± 0.222	0.765 ± 0.208	0.825 ± 0.126
HMC EOPE	0.710 ± 0.178	0.801 ± 0.139	0.842 ± 0.112	0.842 ± 0.115
RMSProp EOPE	0.678 ± 0.274	$\mathbf{0.821} \pm 0.143$	0.855 ± 0.112	$\mathbf{0.863} \pm 0.111$
Deep EOPE	0.696 ± 0.172	0.808 ± 0.140	0.851 ± 0.110	0.842 ± 0.122

Figure 9: Results on Omniglot data set. The first row indicates numbers of original classes selected as negative class, 10 images are sampled from each original class. Greek, Braille and Futurama alphabets are used as normal classes.

relatively well even on high-dimensional problems, which might indicate that gradients produced by its regularization term have variance sufficiently low for a proper convergence.

The main drawback of the OPE and EOPE methods is slow training, which is largely due to the usage of Monte-Carlo methods. It is partially alleviated by fast approximation of Hamiltonian Monte-Carlo and usage of a generator (Kim and Bengio, 2016), and can potentially be improved further, by advanced Monte-Carlo techniques (for example, Levy et al., 2018).

6. Conclusion

We present a new family of anomaly detection algorithms which can be efficiently applied to the problems intermediate between one-class and two-class settings. Solutions produced by these methods combine the best features of one-class and two-class approaches. In contrast to conventional one-class approaches, proposed methods can effectively take into account any number of known anomalous examples, and, unlike conventional two-class classification, does not require a representative sample of anomalous data. Our experiments show better or comparable performance to conventional two-class and one-class algorithms. Our approach is especially beneficial for anomaly detection problems, in which anomalous data is non-representative, or might evolve over time.

Acknowledgments

The research leading to these results has received funding from Russian Science Foundation under grant agreement n° 17-72-20127.

Appendix A. Experimental Setup

In order to make a clear comparison between methods, network architectures are made as close as possible. For image data (MNIST, CIFAR, Omniglot) VGG-like networks (Simonyan and Zisserman, 2014) are used, for tabular data 5-layers dense networks are used, more details can be found in Appendix C[2].

We evaluate the following proposed methods:

- brute-force OPE: described by Algorithm 1;
- HMC EOPE: Algorithm 2 with Hamiltonian Monte-Carlo;
- RMProp EOPE: Algorithm 2 equipped with the pseudo-MCMC defined by Equations (9) and (10);
- Deep EOPE: Algorithm 2 with MCMC sampling procedure replaced by a generator as described by Kim and Bengio (2016);

against the following methods:

2. Implementation can be found at `https://gitlab.com/lambda-hse/ope`.

- Robust AE: a deep AutoEncoder trained as suggested by Chalapathy et al. (2017) with $\lambda = 1$;
- Deep SVDD: a classifier trained as suggested by Tax and Duin (2001);
- semi-supervised: a deep AutoEncoder trained to minimize reconstruction error (MSE) accompanied by a small classifier built on top of the encoder, similar to the first method suggested by Kingma et al. (2014);
- cross-entropy: a classifier trained to minimize cross-entropy.

All OPE and EOPE models are trained with $\varepsilon = 0.95$. All MCMC chains are persistent (by analogy with Tieleman and Hinton, 2009) and 4 MCMC steps are performed for each gradient step. All networks are optimized by Adam algorithm (Kingma and Ba, 2014) with learning rate $5 \cdot 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$.

In order to reflect the assumptions behind the proposed methods, we derive several tasks from each original data set considered. For SUSY, HIGGS and KDD-99 data set positive class is fixed according to data sets' descriptions; for MNIST and CIFAR-10 data sets each class is considered as positive; for Omniglot data set we choose 'Braille', 'Futurama' and 'Greek' alphabets are chosen as positive classes.

In order to fully demonstrate advantages of OPE and EOPE methods we vary sample sizes for negative class: for SUSY and HIGGS data sets only a small number of negative examples is randomly selected (0, 10^2, 10^3, 10^4 and 10^5); for multi-class data sets several classes are randomly selected (without replacement) and subsampled, for MNIST, CIFAR and Omniglot data sets 0, 1, 2 and 4 classes are selected with 10 examples from each, for KDD-99 maximum number of samples per class is limited by 10^3.

Original train-test splits are respected when possible (for SUSY and HIGGS data sets splits are random and fixed for all derived tasks)—test sets are not modified in any way.

Appendix B. Formal Proof of Theorem 1

Here we provide a formal proof of Theorem 1 from the Section 2.2. For the sake of simplicity, we split the proof into two lemmas.

Lemma 1 *Let $(\mathcal{X}, \|\cdot\|)$ be a Banach space, $P(x)$—a continuous probability density function such that $\Omega = \operatorname{supp} P$ is an open set in $\mathcal{X}$. If continuous function $g^* : \Omega \to \mathbb{R}$ minimizes $\mathcal{L}_1^E$ (defined by Equation 6) with $P(x \mid \mathcal{C}^+) = P(x)$, then there exists a strictly increasing function $s : \mathbb{R} \to \mathbb{R}$, such that $g^*(x) = s(P(x))$.*

Proof Consider a continuous function $g : \Omega \to \mathbb{R}$. We show that if g can not be represented as $s(P(x))$, then g does not minimize $\mathcal{L}_1^E$. This is demonstrated by constructing another continuous function g' that achieves lower loss than g.

If g can not be represented as $s(P(x))$ then a pair of points x_1 and x_2 can be found such that $P(x_1) < P(x_2)$ and $g(x_1) \geq g(x_2)$.

Due to continuity of P and g, it is possible to find such neighborhoods of x_1 and x_2, that the difference in probability densities remains large, while differences in values of g become insignificant or negative. More formally, for every $\delta > 0$ there exists $r > 0$ such that open

balls $B_1 = B(x_1, r)$ and $B_2 = B(x_2, r)$, $B_1, B_2 \subset \Omega$ satisfy following properties:

$$\inf_{B_2} P - \sup_{B_1} P > \Delta; \tag{11}$$

$$\sup_{B_2} g - \inf_{B_1} g < \delta; \tag{12}$$

where $2\Delta = P(x_2) - P(x_1)$.

We define function $g'_{\alpha,\beta}$ as $g'_{\alpha,\beta}(x) = g(x) - \alpha\, h(x - x_1) + \beta\, h(x - x_2)$, where $h : \mathcal{X} \to \mathbb{R}$, $\alpha, \beta > 0$; the exact form of h is not important, nevertheless, for clarity, let

$$h(x) = r^{-1} \cdot \max(r - \|x\|, 0). \tag{13}$$

We restrict our attention to such values of α and β, that $g'_{\alpha,\beta}$ has the same normalization constant as g:

$$\Delta Z(\alpha, \beta) = \int_\Omega \exp(g'_{\alpha,\beta}(x))dx - \int_\Omega \exp(g(x))dx = 0. \tag{14}$$

Equation (11) implies that B_1 and B_2 do not intersect and, since $g(x) = g'_{\alpha,\beta}(x)$ for $x \in \Omega \setminus (B_1 \cup B_2)$, $\Delta Z(\alpha, \beta)$ consists of two non-zero terms:

$$\Delta Z(\alpha, \beta) = \Delta Z_1(\alpha) + \Delta Z_2(\beta);$$

where:

$$\begin{aligned} \Delta Z_1(\alpha) &= \int_{B_1} \exp(g(x) - \alpha h(x - x_1)) - \exp(g(x))dx; \\ \Delta Z_2(\beta) &= \int_{B_2} \exp(g(x) + \beta h(x - x_2)) - \exp(g(x))dx. \end{aligned}$$

For every $\alpha \geq 0$ there exist a unique $\beta^*(\alpha) \geq 0$ such that $(\alpha, \beta^*(\alpha))$ is a solution for Equation (14). Notice also, that $\beta^*(\alpha)$ is a continuous, strictly increasing function and $\beta^*(0) = 0$.

Notice, that for small values of α and β

$$\begin{aligned} \Delta Z_1(\alpha) &= \int_{B_1} -\alpha h(x - x_1) \exp(g(x)) + \mathcal{O}(\alpha^2 h^2(x - x_1))dx; \\ \Delta Z_2(\beta) &= \int_{B_2} \beta h(x - x_2) \exp(g(x)) + \mathcal{O}(\beta^2 h^2(x - x_2))dx. \end{aligned}$$

therefore,

$$\lim_{\delta \to 0} \lim_{\alpha \to 0} \frac{\alpha}{\beta^*(\alpha)} \leq 1. \tag{15}$$

Similarly to $\Delta Z(\alpha, \beta)$, $\Delta L^+(\alpha, \beta) = L^+(g'_{\alpha,\beta}) - L^+(g)$ can be split into two parts:

$$\begin{aligned} \Delta L^+(\alpha, \beta) &= \Delta L_1^+(\alpha) + \Delta L_2^+(\beta); \\ \Delta L_1^+(\alpha) &= \int_{B_1} P(x) \left[l(g(x) - \alpha h(x - x_1)) - l(g(x)) \right] dx; \\ \Delta L_2^+(\beta) &= \int_{B_2} P(x) \left[l(g(x) + \beta h(x - x_2)) - l(g(x)) \right] dx; \end{aligned} \tag{16}$$

where $l(y) = \log(1 + \exp(-y))$.

Note, that for a positive Δy

$$\frac{\Delta y}{1 + \exp(y + \Delta y)} < l(y) - l(y + \Delta y) < \frac{\Delta y}{1 + \exp(y)};$$

therefore,

$$\begin{aligned} \Delta L_1^+(\alpha) &< \int_{\|\chi\|<r} \alpha h(\chi) P(x_1 + \chi) J_1(\alpha, \chi) d\chi; \\ \Delta L_2^+(\beta) &< -\int_{\|\chi\|<r} \beta h(\chi) P(x_2 + \chi) J_2(\beta, \chi) d\chi. \end{aligned}$$

where:

$$J_1(\alpha, \chi) = \frac{1}{1 + \exp(g(x_1 + \chi) - \alpha h(\chi))}; \tag{17}$$

$$J_2(\beta, \chi) = \frac{1}{1 + \exp(g(x_2 + \chi) + \beta h(\chi))}; \tag{18}$$

hence,

$$\Delta L^+(\alpha, \beta) < \int_{\|\chi\|<r} h(\chi) \left[P(x_1 + \chi) \alpha J_1(\alpha, \chi) - P(x_2 + \chi) \beta J_2(\alpha, \chi) \right] d\chi.$$

Note, that

$$P(x_1 + \chi) \alpha J_1(\alpha, \chi) - P(x_2 + \chi) \beta J_2(\beta, \chi) \leq \frac{\alpha P_1}{1 + \exp(G_1)} - \frac{\beta P_2}{1 + \exp(G_2 + \beta)} \equiv J(\alpha, \beta);$$

where: $G_1 = \inf_{B_1} g$, $G_2 = \sup_{B_2} g$, $P_1 = \sup_{B_1} P$, $P_2 = \inf_{B_2} P$.

Now, our aim is to prove that $J(\alpha, \beta) \leq 0$ has a solution in form $(\alpha, \beta^*(\alpha))$:

$$J(\alpha, \beta^*(\alpha)) < 0 \Leftrightarrow \frac{\alpha}{\beta^*(\alpha)} < C(\beta^*(\alpha)); \tag{19}$$

where:

$$C(\beta) = \frac{P_2}{P_1} \frac{1 + \exp(G_1)}{1 + \exp(G_2 + \beta)}.$$

Note, that for each $0 < \delta < \log(P_2) - \log(P_1)$, and for each $0 < \beta < \log(P_2) - \log(P_1) - \delta$, $C(\beta) > 1$. In combination with Equation (15), this implies that Inequality (19) is satisfied for some $\alpha > 0$ and $\beta = \beta^*(\alpha)$, therefore, $\Delta Z(\alpha, \beta) = 0$ and $\Delta L^+(\alpha, \beta) < 0$ are simultaneously satisfied for some $\alpha > 0$ and $\beta > 0$. This implies, that function $g'_{\alpha_0, \beta^*(\alpha_0)}$ has the same normalization constant Z as the original one, and reduces value of L^+, hence, g does not minimize $\mathcal{L}_1^E$, which concludes this proof. ■

Lemma 2 *For every function s that satisfies Lemma 1:*

$$\inf_{\Omega} P = 0 \Rightarrow \lim_{y \to 0} s(y) = -\infty.$$

Proof Suppose that $\lim_{y\to 0} s(y) = S \in \mathbb{R}$.

For every sufficiently small $\Delta > 0$, we can pick points $x_1, x_2 \in \Omega$, radius $r > 0$ and two open balls $B_1(x_1, r)$, $B_2(x_2, r)$ such that

$$\begin{aligned} \sup_{B_1} P &< \Delta; \\ \sup_{B_2} P &> 8\Delta. \end{aligned}$$

Now we can introduce the same definitions and constructs as in Lemma 1, applied for B_1, B_2 and g. Consider $\alpha > 0$, such that $\alpha < 2\beta^*(\alpha)$ (such values always exist due to Equation 15). Note that since $\inf_\Omega g \geq S$, for every $\beta > 0$, J_2 (defined by Equation 18) is bounded from below

$$\inf_{\|\chi\|<r} J_2(\beta, \chi) \geq \frac{1}{1 + \exp(S + \beta)}.$$

Consider

$$\Delta = \min\left[\frac{\sup_{\|\chi\|<r} J_1(\alpha, \chi)}{\inf_{\|\chi\|<r} J_2(\beta^*(\alpha), \chi)}, 1\right] \cdot \frac{\sup_{B_2} P}{4}.$$

Such choice of Δ guarantees that for every χ, such that $\|\chi\| < r$,

$$\alpha P(x_1 + \chi) J_1(\alpha, \chi) < \beta^*(\alpha) P(x_2 + \chi) J_2(\beta^*(\alpha), \chi);$$

where J_1 and J_2 are defined by Equations (17) and (18). This makes ΔL^+ from Equation (16) negative, which, in turn, implies that g does not minimize $\mathcal{L}_1^E$, which contradicts our assumptions. ■

Appendix C. Network Architectures

Below we provide exact network architectures used in the experiments. As we aim to match all methods involved in the experiments as close as possible, we introduce several types of networks with the following mapping from methods to network types employed:

- Robust AE—`encoder` → `decoder`;
- Deep SVDD—`classifier` with absent bias terms;
- cross-entropy—`classifier`;
- semi-supervised—`encoder` → `decoder` (unsupervised phase) / `encoder` → `small classifier` (supervised phase);
- OPE methods—`classifier`, Deep EOPE additionally uses `generator`;

`Generator` networks are equivalent to `decoder` ones, with two distinctions:

- former receive samples from a normally distributed random variable as input;
- each layer of a `generator` network is accompanied by a batch normalization layer (Ioffe and Szegedy, 2015) used by Kim and Bengio (2016).

C.1. Notation

Below we specify notation for network description:

- `input(N)`—denotes input layer with N features;
- `random_normal(N)`—denotes N-dimensional normally distributed random variable;
- `dense(N)`—a fully-connected (dense) layer with N output units;
- `conv(N, K, S)` and `deconv(N, K, S)`—a convolution / transposed convolution layer (LeCun et al., 1989) with N K$\times$K filters (with $K = 3$ if unspecified), strides S ($S = 1$ if unspecified) and no padding;
- `max_pool()`, `upscale()`—a max pooling (LeCun et al., 1989) / upscale layer with 2×2 window;
- `global_max_pool()`—computes maximum over all spatial dimensions;

For hidden `dense`, `conv` and `deconv` layers we use leaky Rectified Linear Units (Glorot et al., 2011) with leakiness $\alpha = 0.05$:

$$\mathrm{ReLU}(x) = \begin{cases} x, & \text{if } x > 0, \\ \alpha x, & \text{otherwise;} \end{cases}$$

Output layers does not use any activation function.

C.2. Network Specifications

Tables 10, 11, 12 and 13 provide detailed description of the network architectures employed in the experiments. Note that these architectures are also used for all baseline methods.

classifier	`input(D)` → `dense(4 N)` → `dense(3 N)` → `dense(2 N)` → `dense(N)` → `dense(1)`
encoder	`input(D)` → `dense(4 N)` → `dense(3 N)` → `dense(2 N)` → `dense(N)` → `dense(M)`
decoder	`input(D)` → `dense(N)` → `dense(2 N)` → `dense(3 N)` → `dense(4 N)` → `dense(D)`
small classifier	`input(M)` → `dense(2 N)` → `dense(N)` → `dense(1)`

Figure 10: Specification of the networks used for HIGGS, SUSY and KDD-99 data sets.

For the networks presented in Table 10, we use the following constants:

- $N = 96$ for HIGGS and SUSY data sets, $N = 48$ for KDD-99;
- $M = 14$ for HIGGS, $M = 9$ for SUSY and $M = 62$ for KDD-99 data sets: set to the half of the number of input features;
- $D = 28$ for HIGGS, $D = 18$ for SUSY and $D = 123$ for KDD-99 data sets—the number of input features;

- $K = 64$ for HIGGS and SUSY data sets, $N = 96$ for KDD-99—dimensionality of the generator's latent space.

classifier	input(28×28×1) → conv(32) → conv(48) → max_pool() → conv(64) → conv(96) → max_pool() → conv(128) → global_max_pool() → dense(1)
encoder	input(28×28×1) → conv(32) → conv(48) → max_pool() → conv(64) → conv(96) → max_pool() → conv(128) → global_max_pool() → dense(128)
decoder	input(1×1×128) → upscale() → deconv(96) → upscale() → deconv(64) → deconv(48) → upscale() → deconv(32) → deconv(1)
small classifier	input(64) → dense(64) → dense(32) → dense(1)

Figure 11: Specification of the networks used for MNIST data set.

classifier	input(32×32×3) → conv(32) → conv(48) → max_pool() → conv(64) → conv(96) → max_pool() → conv(128) → conv(128) → global_max_pool() → dense(1)
encoder	input(32×32×3) → conv(32) → conv(48) → max_pool() → conv(64) → conv(96) → max_pool() → conv(128) → conv(192) → global_max_pool() → dense(192)
decoder	input(1×1×192) → upscale() → deconv(128) → deconv(96) → upscale() → deconv(64) → deconv(48) → upscale() → deconv(32) → deconv(1)
small classifier	input(192) → dense(64) → dense(32) → dense(1)

Figure 12: Specification of the networks used for CIFAR-10 data set.

classifier	input(105×105×1) → conv(32, K=5, S=2) → conv(32, K=5, S=2) → conv(48) → conv(48) → max_pool() → conv(96) → conv(96) → max_pool() → conv(128) → global_max_pool() → dense(1)
encoder	input(105×105×1) → conv(32, K=5, S=2) → conv(32, K=5, S=2) → conv(48) → conv(48) → max_pool() → conv(96) → conv(96) → max_pool() → conv(128) → global_max_pool() → dense(128)
decoder	input(1×1×128) → deconv(96) → upscale() → deconv(64) → deconv(64) → upscale() → deconv(48) → deconv(48) → deconv(32, K=5, S=2) → deconv(1, K=5, S=2)
small classifier	input(64) → dense(64) → dense(32) → dense(1)

Figure 13: Specification of the networks used for Omnigot data set.

References

UCI Machine Learning Repository, 1999. URL `https://archive.ics.uci.edu/ml/datasets/kdd+cup+1999+data`.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308, 2014.

Teresa MA Basile, Nicola Di Mauro, Floriana Esposito, Stefano Ferilli, and Antonio Vergari. Density estimators for positive-unlabeled learning. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 49–64, Skopje, Macedonia, 2017.

Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *CoRR*, abs/1811.04820, 2018. URL `http://arxiv.org/abs/1811.04820`.

Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, 2(1):1–127, 2009.

Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Robust, deep and inductive anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 36–51, Skopje, Macedonia, 2017.

Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 08, page 213220, Las Vegas, Nevada, USA, 2008.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, Ft. Lauderdale, Florida, USA, 2011.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on International Conference on Machine Learning*, volume 37 of *ICML15*, page 448456, 2015.

Long Jin, Justin Lazarow, and Zhuowen Tu. Introspective classification with convolutional nets. In *Advances in Neural Information Processing Systems*, pages 823–833, Long Beach, California, United States, 2017.

Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, Montreal, Canada, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Daniel Levy, Matt D. Hoffman, and Jascha Sohl-Dickstein. Generalizing Hamiltonian Monte Carlo with neural networks. In *International Conference on Learning Representations*, Vancouver, Canada, 2018.

Xiao-Li Li and Bing Liu. Learning from positive and unlabeled examples with different data distributions. In *European Conference on Machine Learning*, pages 218–229, Porto, Portugal, 2005.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *IEEE International Conference on Data Mining*, pages 413–422, Washington, DC, United State, 2008.

Curtis G. Northcutt, Tailin Wu, and Isaac L. Chuang. Learning with confident examples: rank pruning for robust classification with noisy labels. In *Conference on Uncertainty in Artificial Intelligence, UAI*, Sydney, Australia, 2017.

Lukas Ruff, Nico Görnitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Robert Vandermeulen, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4390–4399, Stockholm, Sweden, 2018.

Bernhard Scholkopf and Alexander J Smola. *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2001.

Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in Neural Information Processing Systems*, pages 582–588, Denver, United States, 2000.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

David MJ Tax and Robert PW Duin. Uniform object generation for optimizing one-class classifiers. *Journal of machine learning research*, 2(Dec):155–173, 2001.

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Annual International Conference on Machine Learning*, pages 1064–1071, Helsinki, Finland, 2008.

Tijmen Tieleman and Geoffrey Hinton. Using fast weights to improve persistent contrastive divergence. In *Annual International Conference on Machine Learning*, pages 1033–1040, Montreal, Canada, 2009. ACM.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

Zhuowen Tu. Learning generative models via discriminative approaches. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, Minnesota, United States, 2007.

Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674, Halifax, Canada, 2017.