PeerJ Computer Science

A hybrid anomaly detection method for high dimensional data

Xin Zhang[1], Pingping Wei[1] and Qingling Wang[2]

[1] School of Intelligent Science and Engineering, Yunnan Technology and Business University, Kunming, China
[2] Chongqing Technology and Business Institute, Chongqing, China

ABSTRACT

Anomaly detection of high-dimensional data is a challenge because the sparsity of the data distribution caused by high dimensionality hardly provides rich information distinguishing anomalous instances from normal instances. To address this, this article proposes an anomaly detection method combining an autoencoder and a sparse weighted least squares-support vector machine. First, the autoencoder is used to extract those low-dimensional features of high-dimensional data, thus reducing the dimension and the complexity of the searching space. Then, in the low-dimensional feature space obtained by the autoencoder, the sparse weighted least squares-support vector machine separates anomalous and normal features. Finally, the learned class labels to be used to distinguish normal instances and abnormal instances are outputed, thus achieving anomaly detection of high-dimensional data. The experiment results on real high-dimensional datasets show that the proposed method wins over competing methods in terms of anomaly detection ability. For high-dimensional data, using deep methods can reconstruct the layered feature space, which is beneficial for gaining those advanced anomaly detection results.

Subjects Algorithms and Analysis of Algorithms, Data Mining and Machine Learning, Data Science
Keywords Anomaly detection, Autoencoder, High-dimensional data, Support vector machine

Submitted 26 July 2022
Accepted 5 December 2022
Published 12 January 2023

Corresponding author
Xin Zhang, jsjgcxz@163.com

Academic editor
Carlos Fernandez-Lozano

Additional Information and Declarations can be found on page 14

DOI **10.7717/peerj-cs.1199**

INTRODUCTION

Anomaly detection is an important part of data mining. There are many cases regarding anomaly appearance, such as system failure, abnormal behavior, data outliers, *etc.* Anomalies (aka outliers) have properties: (i) rarity, *i.e.,* it is difficult to label anomalies because anomaly instances are sparse; and (ii) type diversity, *i.e.,* there are many types of anomalies, such as, point anomalies, group anomalies, conditional anomalies, *etc.*

Anomalous features are more likely to be manifested in a low-dimensional space, but they are more hidden in a high-dimensional space. Due to high dimensionality, the distance contrast between data becomes similar (*Yu & Chen, 2019*; *Menon & Kalyani, 2019*), whereas, most anomaly detection methods explicitly or implicitly rely on distance contrast (*Li, Lv & Yi, 2020*). Obviously, high dimensionality easily leads to fail in anomaly detection for these methods relying on distance contrast. Furthermore, the data in a high-dimensional space presents a sparse distribution, which difficult affords rich information distinguishing abnormal and normal instances (*Soleimani & Miller, 2016*). In this case, anomaly detection for the data in a high-dimensional space is a challenge.

Currently, anomaly detection methods allow to be divided into the following categories: (i) Distance metric-based methods do not acquire data distribution, *e.g.*, K-nearest neighbor (K-NN) (*Chehreghani, 2016*), Random Distances (*Wang et al., 2020b*). Distance metrics become more and more similar along with the increasing of data dimensionality, so such methods easily suffer negative effects of high dimensionality. (ii) Deep learning-based methods can not only learn deep features of the data, but also interpret the learned features (*Yuan et al., 2018*), *e.g.*, Bayesian Variational Autoencoder (BVAE) (*Daxberger & Hernández-Lobato, 2019*), and these methods in *Grosnit et al. (2022)*, *Bourached et al. (2022)*, *Grathwohl et al. (2019)*, *Goodfellow, Bengio & Courville (2019)* and *Ian et al. (2014)*. Such detection methods have unsupervised detection methods and supervised detection methods, where unsupervised detection methods do not rely on data labels but they are very sensitive to noise and missing data (*Parchami et al., 2017*), *e.g.*, Deep One-class Classification (DOC) (*Ruff et al., 2018*), Generative Adversarial Network (GAN) (*Li et al., 2019*), *etc*. The objective function of unsupervised detection methods is more used for data dimensionality reduction or data compression, so anomaly detection accuracy may be usually lower than that of supervised detection methods. Unlike unsupervised detection methods, Supervised detection methods show better detection performance because of relying on data labels, such as these methods in *Metzen et al. (2017)* and *Grosse et al. (2017)*. Unfortunately, labeling the data is a challenge if the amount of data is large or the data is multi-dimensional. Therefore, supervised anomaly detection methods are difficult to adapt to anomaly detection of high-dimensional data and large-scale data. (iii) Deep hybrid-based methods, such as Deep Neural Networks-Support Vector Machine (DNN-SVM) (*Inoue et al., 2017*), deep autoencoder and ensemble k-nearest neighbor (DAE-KNN) (*Song et al., 2017*), consist of deep methods and traditional detection methods; therefore, they inherit the characteristics of deep detection methods and traditional detection methods. Meanwhile, they own natural advantages in anomaly detection, but there needs trade-off computational cost and calculation accuracy. A typical representative of (iv) traditional detection methods are the support vector machine (SVM)-based methods, such as OC-SVM (*Ergen, Hassan Mirza & Serdar Kozat, 2017*), SVM (*Erfanin et al., 2016*; *Shi & Zhang, 2021*). SVM-based methods are susceptible to the linear inseparability of high-dimensional data (*Jerónimo Pasadas et al., 2020*); therefore, *Wang et al. (2020a)* proposed the improved SVM-based method.

The motivation of this article is to detect anomalies of high-dimensional data. Here, this article proposes a hybrid approach combining an autoencoder and a sparse weighted least squares support vector machine, namely AE-SWLS-SVM. Firstly, the autoencoder is used to extract low-dimensional features of high-dimensional data, thereby reducing data dimensionality and the complexity of the searching space. Then, in the low-dimensional feature space obtained by the autoencoder, the sparse weighted least squares support vector machine separates normal and abnormal features. Finally, the class labels being used to distinguish normal instances and abnormal instances are sent out, thereby realizing anomaly detection of high-dimensional data.

We summarize main contributions of this works.

(I) The proposed AE-SWLS-SVM can adapted well to high-dimensional environments during anomaly detection. Since the autoencoder captures the layered features from high-dimensional data, which provides beneficial environments for the sparse weighted least squares-SVM to distinguish the normal features and abnormal features.

(II) For high-dimensional data, the layered feature space reconstructed by deep methods is beneficial for gaining those advanced anomaly detection results. The contrast of distance between the data becomes difficulty as data dimensionality increases in high-dimensional spaces, however, in the layered feature space reconstructed by deep methods, the contrast of distance between the data becomes significant.

MATERIALS AND METHODS

Overall scheme

Figure 1 displays the overall scheme of the proposed method, which includes feature extraction, feature separation and instance reconstruction. In the first stage, namely feature extraction, the encoder captures low-dimensional features from the input data, which provides good environments for feature separation in the next stage. In the second stage, *i.e.,* feature separation, the sparse weighted least squares-support vector machine achieves the separation of abnormal and normal features in the low-dimensional feature space. In the third stage, namely instance reconstruction, the decoder reconstructs normal and abnormal instances from the separated normal and abnormal features. Finally, the learned class labels are output.

Feature extraction

Autoencoders show excellent ability in capturing low-dimensional features of high-dimensional data. For simplicity, $\Re^h \xrightarrow{\Omega} \Re^l$ denotes that the autoencoder extracts low-dimensional features from high-dimensional data, where $\Re^h$ represents an h-dimensional high-dimensional space, and the data dimension in the space is h dimensionality. Similarly, $\Re^l$ represents the corresponding low-dimensional feature space, and the dimensionality is l, and $l < h$. Ωrepresents our autoencoder, which consists of an input layer, an output layer and multiple hidden layers. The formal description for Ω is as follow.

Input layer In_Ωachieves the mapping of input data, *i.e.,* the data in $\Re^h$ ismapped onto In_Ω.

Multiple hidden layers. The encoder and the decoder contain hidden layers, respectively, so we need to describe them, respectively.

(i) Hidden layers in the encoder. The input and the output of the n-th hidden layer in the i-th iteration are denoted as $E^{\text{in}}_{n,i}, E^{\text{out}}_{n,i}$, respectively. $E^{\text{out}}_{n,i}, E^{\text{in}}_{n,i}$ are calculated by Eqs. (1) and (2).

$$E^{\text{out}}_n = \nabla^e_n(w^i_n E^{\text{in}}_{n,i} + b^i_n) \tag{1}$$

$$E^{\text{in}}_{n,i} = E^{\text{out}}_{n-1,i} \tag{2}$$

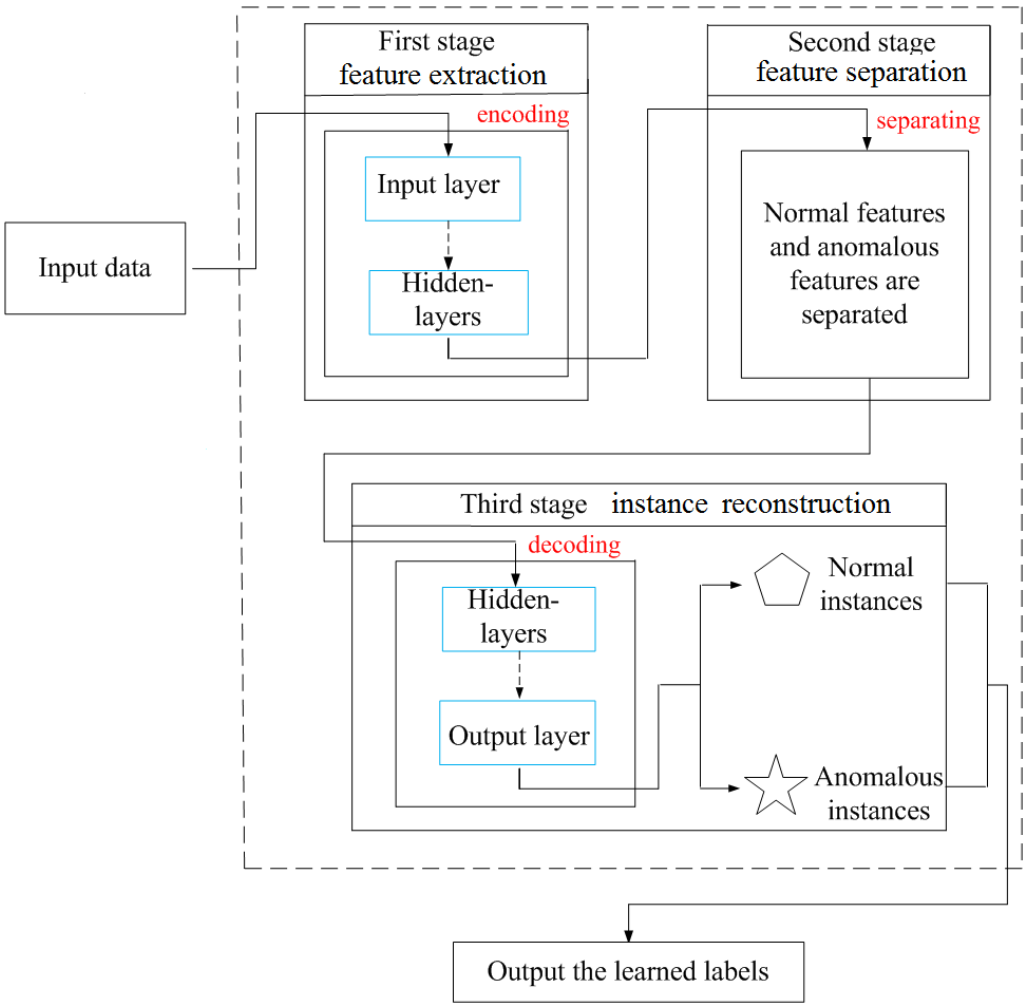

Figure 1 **Overall scheme.**

Full-size DOI: 10.7717/peerjcs.1199/fig-1

(ii) Hidden layers in the decoder. Correspondingly, the input and the output of the m-th hidden layer in the i-th iteration are denoted as $D^{\text{in}}_{m,i}, D^{\text{out}}_{m}$, respectively, as following,

$$D^{\text{out}}_{m} = \Delta^{d}_{m}(w^{i}_{m} D^{\text{in}}_{m,i} + b^{i}_{m}) \tag{3}$$

$$D^{\text{in}}_{m,i} = D^{\text{out}}_{m-1,i} \tag{4}$$

where $\nabla^{e}_{n}, \Delta^{d}_{m}$ are activation function in coding hidden layers and decoding hidden layers, respectively. **w** and **b** weight in hidden layers and bias.

Output layers out_{Ω} sends out the reconstructed instances. Ω isformally defined as follow

$$x \underset{\text{input}}{\rightarrow} In_{\Omega} \xrightarrow{E^{\text{in}}_{n,i} \ldots E^{\text{out}}_{n,i}, D^{\text{in}}_{m,i}, \ldots, D^{\text{out}}_{m}} \text{out}_{\Omega} \underset{\text{output}}{\rightarrow} z \tag{5}$$

where x, z are the input and the reconstructed input, respectively. The loss function L_Ω of Ω is given in Eq. (6)

$$L_\Omega = ||x - z||^2 \tag{6}$$

Indeed, proper regularization can improve the ability of autoencoders to capture features (*Lu et al., 2017*), therefore, L_Ω is regularized by introducing regularization item and J–S(Jensen–Shannon) divergence JS_{sparse}, having that

$$L_\Omega = ||x - z||^2 + L_2 + JS_{\text{sparse}} \tag{7}$$

where L_2 (*Olshausen & Field, 1997*) is the regularization item and optimizes the weight of Ω in order to ensure that the components of $\mathbf{w}$ are as balanced as possible. JS_{sparse} divergence is a variant based on K–L divergence. Because JS_{sparse} divergence is symmetric, it can solve the problem of K-L divergence asymmetry (*Cattai et al., 2021*; *Li et al., 2021*). JS_{sparse} divergence calculation formula is as follows

$$\begin{cases} JS_{\text{sparse}}(P_1||P_2) = \frac{1}{2}[KL(P_1||\frac{P_1+P_2}{2}) + KL(P_2||\frac{P_1+P_2}{2})] \\ KL(P_1||\frac{P_1+P_2}{2}) = -\sum_{x\in X} P_1(x)\log\frac{1}{P_1(x)} + \sum_{x\in X} P_1(x)\log\frac{1}{P_2(x)} \\ KL(P_2||\frac{P_1+P_2}{2}) = -\sum_{x\in X} P_2(x)\log\frac{1}{P_2(x)} + \sum_{x\in X} P_2(x)\log\frac{1}{P_1(x)} \end{cases} \tag{8}$$

where P_1 represents the true distribution of the data. P_2 is the theoretical distribution of the data or an approximate distribution of P_1.

Feature separation

Support Vector Machine (SVM) is often used for classification tasks because of excellent classification ability. Anomaly detection can be thought as a binary-classification of normal and abnormal classes. Based on this, we improved the structure of SVM, *i.e.*, sparse weighted least squares was implemented to SVM, denoted as SWLS-SVM.

Given the input sample $\{(\hat{x}_i, \hat{y}_i)|i = 1, 2, .., n\}$, $\hat{y}_i$ is the label of $\hat{x}_i$. SWLS-SVM is defined as following

$$\left.\begin{aligned} &\min \frac{1}{2}||w||^2 + \frac{1}{2}\lambda\sum_{i=1}^{n}\beta_i\xi_i^2 \\ &L_i = w^T\phi(\hat{x}_i, \hat{y}_i) + b + \xi_i \end{aligned}\right\} \tag{9}$$

where w and b are weight and bias. λ is a regularization parameter. β_i is weight coefficient. ξ_i and L_i are an error item and the error function, respectively. $\phi(\bullet)$ is a non-linear mapping function. The goal of SWL-SVM is to minimize the $\frac{1}{2}||w||^2 + \frac{1}{2}\lambda\sum_{i=1}^{n}\beta_i\xi_i^2$.

SWLS-SVM employs a structural risk model, including empirical risk $\frac{1}{2}\lambda\sum_{i=1}^{n}\beta_i\xi_i^2$ and the regularisation term $\frac{1}{2}||w||^2$. If the regularization parameter λ is larger, then the empirical risk becomes more important, certainly, this easily lead to over-fitting. If the regularization parameter λ is small, then the empirical risk becomes less important, so that the effect of anomalies on the function L_i can be ignored.

Equation (9) is a convex quadratic programming problem, and the dual problem can be obtained by using Lagrange multipliers. The constrained objective function in Eq. (9) can be transformed into an unconstrained Lagrangian function, having that

$$F(w,b,\xi_i)=\frac{1}{2}||w||^2+\frac{1}{2}\lambda\sum_{i=1}^{n}\beta_i\xi_i^2-\sum_{i=1}^{n}\alpha_i(w^T\phi(\hat{x}_i,\hat{y}_i)+b+\xi_i-L_i) \tag{10}$$

where $\alpha_i>0$ is the Lagrange multiplier. To minimize $F(w,b,\xi_i)$, we set the partial derivatives of w, b, ξ_i,α_i to zero, respectively.

$$\left.\begin{aligned}
&\frac{\partial F(w,b,\xi_i)}{\partial w}=0\rightarrow w=\sum_{i=1}^{n}\alpha_i\phi(\hat{x}_i,\hat{y}_i)\\
&\frac{\partial F(w,b,\xi_i)}{\partial b}=0\rightarrow \sum_{i=1}^{n}\alpha_i=0\\
&\frac{\partial F(w,b,\xi_i)}{\partial \xi_i}=0\rightarrow \alpha_i=\lambda\beta_i\xi_i\\
&\frac{\partial F(w,b,\xi_i)}{\partial \alpha_i}=0\rightarrow w^T\phi(\hat{x}_i,\hat{y}_i)+b+\xi_i-L_i=0
\end{aligned}\right\}. \tag{11}$$

The mapping function $\phi(\bullet)$ in Eq. (11) can be converted into a kernel function $\kappa(\bullet)$ according to the KKT(Karush-Kuhn–Tucker) (*Peng & Xu, 2013*) condition, having that

$$\kappa((\hat{x}_i,\hat{y}_i),(\hat{x}_j,\hat{y}_j))=\phi(\hat{x}_i,\hat{y}_i)^T\phi(\hat{x}_j,\hat{y}_j) \tag{12}$$

Let Eq. (12) be taken into Eq. (11), the linear Eq. (13) can be obtained by eliminating the variables w and ξ_i, as follows

$$\begin{bmatrix}
0 & 1 & \vdots & 1\\
1 & \kappa((\hat{x}_1,\hat{y}_1),(\hat{x}_1,\hat{y}_1))+1/(\lambda\beta_1) & \vdots & \kappa((\hat{x}_1,\hat{y}_1),(\hat{x}_n,\hat{y}_n))\\
\vdots & \vdots & \vdots & \vdots\\
1 & \kappa((\hat{x}_n,\hat{y}_n),(\hat{x}_1,\hat{y}_1)) & \vdots & \kappa((\hat{x}_n,\hat{y}_n),(\hat{x}_n,\hat{y}_n))+1/(\lambda\beta_n)
\end{bmatrix}\bullet\begin{bmatrix} b\\ \alpha_1\\ \vdots\\ \alpha_n\end{bmatrix}=\begin{bmatrix} 0\\ L_1\\ \vdots\\ L_n\end{bmatrix} \tag{13}$$

By solving Eq. (13), the bias b and the Lagrange multiplier α_i can be obtained.

$$f(\hat{x},\hat{y})=\sum_{i=1}^{n}\alpha_i\kappa((\hat{x},\hat{y}),(\hat{x}_i,\hat{y}_i))+b \tag{14}$$

Any semi-positive definite symmetric function can be used as a kernel function (*Jayasumana et al., 2014*), therefore, we obtain a positive definite function through calculating the cumulative distribution function (*Jayasumana et al., 2013*), having

$$\kappa(x_1,x_2)=(1-(1-x_1^A)^B,1-(1-x_2^A)^B|A,B) \tag{15}$$

where A, B are the non-negative kernel parameters.

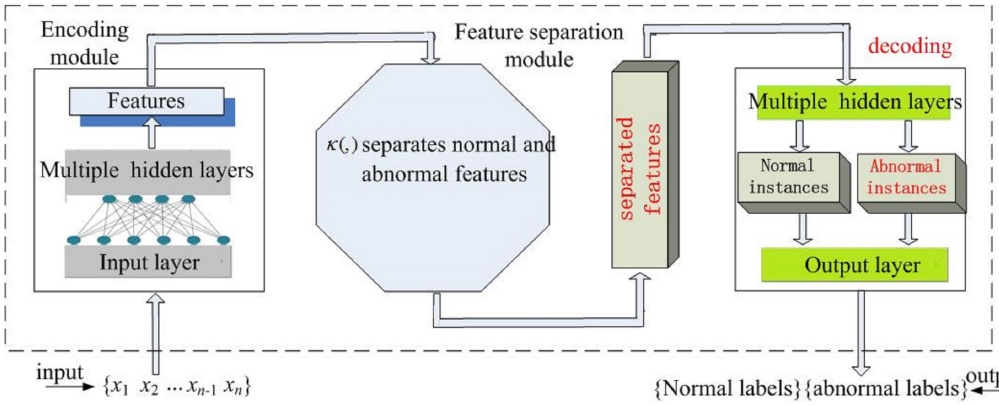

Figure 2 **Model structure.**

Full-size DOI: 10.7717/peerjcs.1199/fig-2

MODEL

Model structure and training

The proposed AE-SWLS-SVM consists of encoding module, feature separation module and decoding module, as shown in Fig. 2. The role of each module is described below.

Module 1, *i.e.,* encoding module. In the module, hidden layers capture the low-dimensional features of the input data. The error of capturing low-dimensional features can be minimized by Eq. (7).

Module 2, *i.e.,* feature separation module. The module separates normal and abnormal features. In the low-dimensional feature space, the kernel function in Eq. (15) completes the separation of normal features and abnormal features.

Module 3, *i.e.,* decoding module. According to the separated features of the output, hidden layers in the module reconstruct normal and abnormal instances. Finally, the input layer outputs the learned normal and abnormal class labels.

The objective function of AE-SWLS-SVM consists of the loss function L_Ω of the autoencoder in Eq. (7) and the error function $f(\hat{x},\hat{y})$ in Eq. (14) of SWLS-SVM, having that

$$T_{\text{target}} = \min\{L_\Omega + f(\hat{x},\hat{y})\}. \tag{16}$$

The performance of AE-SWLS-SVM is related to parameters, so parameters of the model need to be cross-validated in advance, as shown in Algorithm 1. Firstly, the training set is used to train the model, and then the testing set is used to validate the trained model. By analyzing the testing results, the optimal kernel parameter value and the number of neurons are selected. The procedure in Step 2 to Step 18 mainly performs cross-validation of the number of neurons to obtain the optimal number of neurons $opt(\delta)$, where the procedure between Step 3 and Step 14 is the cross-validation of the kernel parameter to obtain the optimal kernel parameter value $_{opt(\gamma)}$.

After obtaining the optimal parameters, we use the training set to train AE-SWLS-SVM, as shown in Algorithm 2. The process of Step 1 to Step 9 realizes the training of the

model. During the training, the objective function is iteratively calculated, when the model converges, we stop the training of the model and save the current training accuracy. From Step 10 to Step 14, we select the maximum trainingaccuracy in the t_{max}-th training as the final output accuracy of the model, and then save the model for the t_{max}-th training.

Algorithm 1. Cross-validation of parameters.

Input: iteration epoch T, the number of neurons δ_1, δ_2, constant $\Delta\delta$, kernel parameter sets γ_k, training set Train_set, testing set Test_set.

Output:

Optimal kernel parameter value $_{opt(\gamma)}$, optimal number of neurons $opt(\delta)$.

Begin

1 **for** $t = 1$ **to** T **do**:
2 **for** $_{\delta=\delta_1}$ **to** $_{\delta_2}$ **do**:
3 **foreach** γ **in** γ_k:
4 Train model SWLS-SVM(Train_set, δ, γ);
5 **for** $i = 1$ **to** I **do**:
6 Calculate the weight coefficient β_i;
7 Learn objective function T_{target};
8 Calculate training accuracy $Train\_Acc$ = SWLS-SVM(Train_set, δ, γ) ;
9 **end for**
10 Test model SWLS-SVM(Test_set, δ, γ) ;
11 Calculate testing accuracy $Test\_Acc$ = SWLS-SVM(Test_set, δ, γ);
12 **end foreach**
13 Select the optimal γ so that maximize testing accuracy $\gamma(\max)$ = argmax($Test\_Acc(\gamma_{\max})$);
14 Obtain optimal kernel parameter $_{opt(\gamma)=\gamma(\max)}$;
15 $\delta = \delta + \Delta\delta$;
16 **end for**
17 Select the optimal δ so that maximize testing accuracy $\delta(\max)$ = argmax($Test\_Acc(\delta_{\max}, \gamma_{\max})$);
18 Obtain the optimal number of neurons $opt(\delta) = \delta(\max)$;
19 **end for**

End

Algorithm 2. Model training.
Input: iteration epoch T, $_{opt(\gamma)}$, $opt(\delta)$, training set Training_set.
Output: training accuracy *Training_acc*.
Begin
for t =1 **to** T **do**:
Train model SWLS-SVM(Training_set; $_{opt(\gamma)}$; $opt(\delta)$) ; style=padding-left:1pc;,
for i =1 **to** I **do**:
Calculate the weight coefficient β_i;
Learn objective function T_{target};
Calculate training accuracy $Training\_acc\ (t) =$ SWLS-SVM(Training_set; $_{opt(\gamma)}$) ;
Save the t-th training accuracy $T\ \_acc\ (\text{save}(t)) = Training\_acc\ (t)$;
end for
end for
traverse the saved training accuracy $T\ \_acc\ (\text{save}(t))$;
Select the maximum training accuracy in the t_{max}-th training
$t_{max} = \arg\max(T\ \_acc\ (\text{save}(t)))$;
Save the model in the t_{max}-th training $_{\text{Save=SWLS-SVM(Training\_set;}opt(\gamma);opt(\delta);t_{\max})}$;
Output training accuracy in the t_{max}-th $Training\_acc = Training\_acc\ (t_{max})$;
End

Model parameters

In order to better train AE-SWLS-SVM, the parameters that have a significant impact on the training results are investigated.

(1) Kernel parameter. To induce the best performance of AE-SWLS-SVM, the optimal value of kernel parameter is searched within a certain range.

(2) Weight coefficient β_i is calculated as follows,

$$\beta_i = \begin{cases} 1, |\dfrac{\xi_i}{\bar{s}}| \geq 1 \\ |\dfrac{\xi_i}{\bar{s}}| |\dfrac{\xi_i}{\bar{s}}| < 1 \\ 10^{-7} |\dfrac{\xi_i}{\bar{s}}| = 0 \end{cases} \tag{17}$$

where $\bar{s}$ isthe standard deviation of ξ_i.$|\xi_i/\bar{s}|$ indicates the absolute value of the error rate.

(3) Activation function. Activation functions used in machine learning are usually Sigmoid, tanh, ELU, ReLU, etc. The output of Sigmoid is only 0 and 1, which is suitable for the judgment of normal instances and abnormal instances. Therefore, Sigmoid is used as the activation function for the proposed AE-SWLS-SVM.

(4) Optimizer and learning rate. Adam is used as an optimizer for AE-SWLS-SVM. Adam not only possesses AdaGrad's ability to handle sparse gradients, but also has RMSProp's ability to handle non-stationary targets (*Kingma & Lei Ba, 2015*). Moreover, Adam can provide different adaptive learning rates for different hyper parameters.

(5) Number of neurons. The number of neurons was determined by cross-validation. Given that data dimension and data volume of the input data, using a certain range to configure the number of neurons can improve the ability of the model to resist over-fitting.

(6) Iteration epoch. During the training of AE-SWLS-SVM, we observe the change of training accuracy and dynamically adjust iteration epoch until the model can converge, and then stop training.

EXPERIMENTS

Datasets

In practical applications, real high-dimensional anomaly datasets are difficult to be obtained. Therefore, we selected real high-dimensional datasets of being often used for classification, and then used the method (*Campos et al., 2016*) to convert these high-dimensional datasets into anomaly detection datasets. We considered seven high-dimensional classification datasets (*i.e.,* U1–U7, the dimensions are greater than 165) to test the anomaly detection ability of the model. Then, we randomly selected five high-dimensional datasets from U1–U7 as the training set to train our model, and the selected process was repeated five times independently. In addition, three benchmark datasets (namely B1, B2, B3) were also selected, after our model is well trained, benchmark dataset B1, B2, B3 are used for parametric cross-validation and model structure validation. Table 1 gives a detailed description of these 10 datasets (seven high-dimensional datasets and three benchmark datasets).

Assessment metrics

Accuracy and F1-score are used as evaluation indicators, and the calculation formula is as follows

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{18}$$

$$\text{F1-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{19}$$

where TP represents the number of correctly predicted anomalous instances. TN represents the number of correctly predicted normal instances. FP represents the number of normal instances that are predicted to be anomalous instances. FN represents the number of anomalous instances that are predicted to be normal instances.

Comparison methods

The proposed method is a hybrid method based on deep network architectures, so deep hybrid-based methods, *i.e.,* DNN-SVM (*Inoue et al., 2017*), DAE-KNN (*Song et al., 2017*), and deep networks-based methods, *i.e.,* GAN (*Li et al., 2019*), are used for comparisons. In addition, distance metric-based methods, *i.e.,* K-NN (*Chehreghani, 2016*) are also considered.

As for the proposed method, the adjustment range of the kernel parameter is defined as $\gamma_k = \{0.1, 0.3, 0.5, 1, 2, 3, 5\}$, the number of neurons $_\delta = [_{\delta_1}, _{\delta_2}]$, $\delta_1 = 10$, $\delta_2 = 110$, and step $\Delta\delta = 20$. To have a fair comparison, the comparison method adopts the parameters observed in the corresponding literature.

Table 1 Dataset information.

#	Benchmark datasets	Description (normal *vs.* anomaly)	Data volume		Anomaly ratio	Data dimension
			Normal	Anomaly		
B1	Shuttle	Class '1' vs. Others	1,000	13	1.28%	9
B2	PenDigits	Other vs. Class '4'	9,868	20	0.20%	16
B3	Waveform	Others vs. Class '0'	3,343	100	2.9%	21

#	High-dimensional datasets	Description (normal *vs.* outliers)	Data volume		Anomaly ratio	Data dimension
			Normal	Anomaly		
U1	Arcene	Normal patterns vs cancer	8,459,427	540,573	6.01%	10,000
U2	Gisette	Zero vs non-zero values	28,278,760	4,221,240	12.99%	5,000
U3	Micro Mass	Zero vs non-zero values	464,819	3,181	0.68%	1,300
U4	Malware	Zero vs non-zero values	2,894,954	37,772	1.29%	1,087
U5	CNAE	Zero vs non-zero values	918,537	7,023	0.76%	857
U6	Epileptic Seizure	eizure vs. non-seizure	11,321	179	1.56%	179
U7	Musk	Musk vs non-musk	79,699	269	0.34%	168

We implemented corresponding algorithms of these methods using Python 3.8 on the Tensorflow 2.0 on Linux System. The environment setting are Intel i7 3.0 GHz CPU, and 32G memory. These algorithms run on the same GPU, and adopt the same configuration.

RESULTS AND DISCUSSION

Parameter testing of the model

AE-SWLS-SVM contains multiple hidden layers and is tested in the range {1, 2, 3, 5, 7, 10, 20, 30 }. The seven high-dimensional datasets are used as experiment datasets, *i.e.,* U1, U2, U3, U4, U5, U6, U7, we randomly selected five high-dimensional datasets from U1-U7 as the training set to train our AE-SWLS-SVM; meanwhile, the selected process was repeated five times independently. After our AE-SWLS-SVM is well trained, the benchmark dataset B1 is used as the testing set to test the number of hidden layers of AE-SWLS-SVM.

Testing results show that when the number of hidden layers reaches 3, the detection performance (including Accuracy and F1-score) of AE-SWLS-SVM tends to be stable, as shown in Fig. 3A. This shows that the proposed model is stable on the cases considered.

Let the number of hidden layers be equal to 3, then parameter are tested on the benchmark dataset B2 and B3, respectively, as shown in Figs. 3B and 3C. Results show that the detection performance of AE-SWLS-SVM is the best when kernel parameter γ is equal to 0.5 and the number of neurons δ is equal to 50. Therefore, in the subsequent experiments, the parameters of AE-SWLS-SVM are configured as the number of hidden layers is 3, and $\gamma = 0.5, \delta = 50$.

Performance comparison

Results on datasets U1-U7 show that AE-SWLS-SVM outperforms competitors DNN-SVM, DAE-KNN, GAN and K-NN in terms of anomaly detection performance on most datasets,

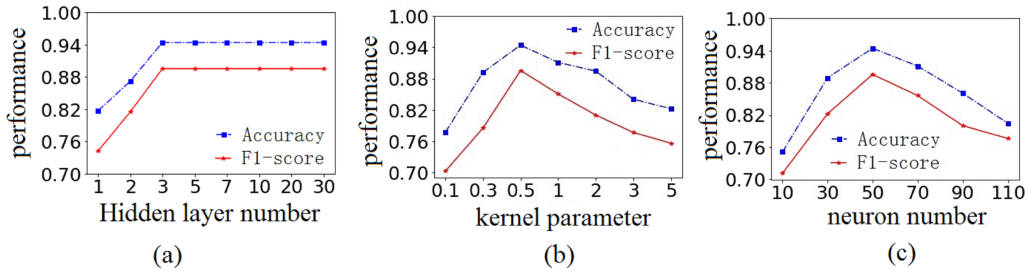

Figure 3 **Testing on benchmark dataset B1, B2 and B3.** Thesting results on benchmark dataset (A) B1, (B)2 and (C)3, respectively.

Full-size DOI: 10.7717/peerjcs.1199/fig-3

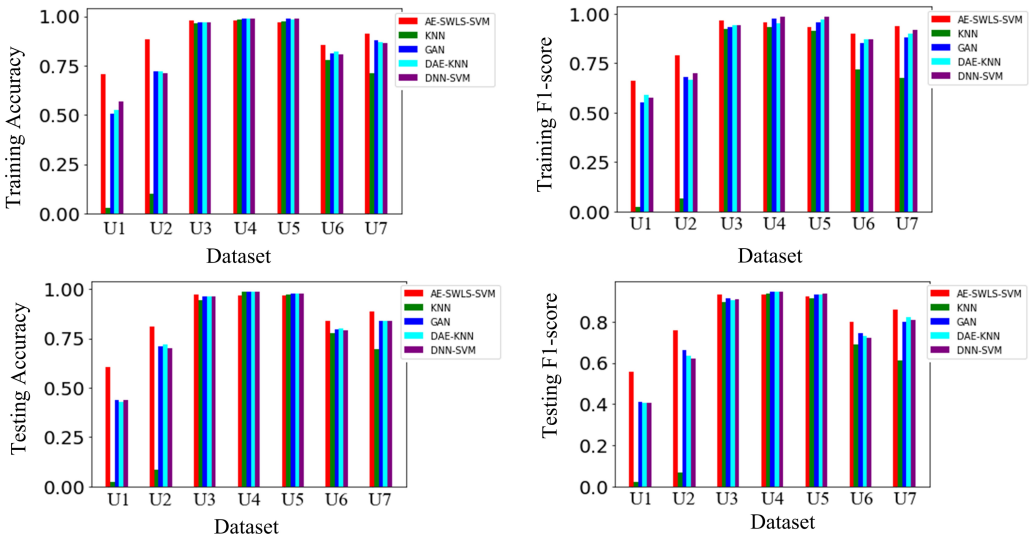

Figure 4 **Detected results.** These methods are compared in the Training/Testing Accuracy metric and Training/Testing F1-score metric.

Full-size DOI: 10.7717/peerjcs.1199/fig-4

as shown in Fig. 4. On higher dimensional datasets, such as U1 (dimension = 10,000) and U2 (dimension = 5,000), anomaly detection advantages of AE-SWLS-SVM are very significant. Unfortunately, the competitor K-NN almost fails on datasets U1 and U2. This is because the contrast of distance between the data becomes more difficulty as data dimensionality increases in high-dimensional spaces, unfortunately, the measurement manner based on distance may difficulty measures the attribute similarity of high-dimensional data. In fact, K-NN based on distance methods is prone to rely on measurement of distance. Although the detection results of these four competitors are better than our AE-SWLS-SVM on datasets U4 and U5, the advantages are not significant. Overall, AE-SWLS-SVM shows more advantages for anomaly detection on high-dimensional data.

Discussion

Insights

Compared with the above competitors, our model has outstanding advantages in terms of anomaly detection for the following reasons.

The autoencoder can capture the low-dimensional layered features from the input data, which is crucial because they provide a sufficient condition for the separation both anomaly features and normal features. The loss function of the autoencoder in Eq. (7) can minimize the error of the extracted low-dimensional layered features. This provides good environments for the kernel in Eq. (15) separating anomaly features from normal features. Through iteratively learning the objective function in Eq. (16), our model can gain good detection accuracy for anomalies.

For these anomaly detection methods, (i) distance metric-based detection methods, such as K-NN (*Chehreghani, 2016*), can work in low-dimensional spaces well, while their detection capabilities are restricted because the contrast between data points in high-dimensional spaces becomes similar, while (ii) deep learning-based detection methods, *e.g.*, GAN (*Li et al., 2019*), are suitable for working upon complex high-dimensional spaces due to owning nonlinear layers extracting important features or learning useful representations. Certainly, in terms of Generative Adversarial Networks (GANs), there exists an unavoidable mode collapsing so that the training is not easy. In regard to (iii) deep hybrid-based detection methods, such as DNN-SVM (*Inoue et al., 2017*), DAE-KNN (*Song et al., 2017*), the methods more and more popular since they inherit the advantages of deep network architectures and traditional detection methods. However, deep hybrid detection methods also show poor detection capabilities when traditional detection methods depend on data distribution or easily occur over-fitting, for example, DBN-Random Forest (*Kam Ho, 1995*) shows poor noise resistance and encounters high risk of over-fitting, due to random forest easily suffers over-fitting on the sample with relatively large noise (*Zheng & Zhao, 2020*; *Popolin Neto & Paulovich, 2021*).

Limitations

The detection performance of the proposed method relies on the extracted features, which means that the quality of the extracted feature has important effects on the ability of the method. Additionally, due to lacking real anomaly datasets, the detection accuracies of most anomaly detection methods are restricted so that it is difficult to truly reflect the detection capabilities of them.

CONCLUSION

This article proposes a hybrid method of combining an autoencoder and a sparse weighted least squares support vector machine for anomaly detection on high-dimensional data. The key thought is that the autoencoder extracts the low-dimensional layered features from high-dimensional data, in order to reduce the dimension of the data and the complexity of the searching space. In the low-dimensional feature space, the sparse weighted least squares-support vector machine separates anomalous features from normal features. Finally, the class labels of being used to distinguish normal instances and abnormal

instances are output, thereby completing anomaly detection of high-dimensional data. Results show that the proposed method is superior to competitors in terms of anomaly detection ability of high-dimensional data. In future works, we will look at addressing the issue of anomaly detection of noise interference. Noise can mask the rare anomalies so that anomalies are likely to be mistaken as noise.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

The authors received no funding for this work.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Xin Zhang conceived and designed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the article, and approved the final draft.
- Pingping Wei performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Qingling Wang performed the experiments, analyzed the data, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The data and code is available in the Supplemental Files.

All datasets used in this work can be found at the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/.

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.1199#supplemental-information.

REFERENCES

Bourached A, Griffiths R-R, Gray R, Jha A, Nachev P. 2022. Generative model-enhanced human motion prediction. *Applied AI Letters* **3**(**2**):1–20.

Campos GO, Zimek A, Sander J, Campello RJGB, Micenková B, Schubert E, Assent I, Houle ME. 2016. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. *Data Mining & Knowledge Discovery* **30**:891–927 DOI 10.1007/s10618-015-0444-8.

Cattai T, Scarano G, Corsi M-C, Bassett DS, De Vico Fallani F, Colonnese S. 2021. Improving J-divergence of brain connectivity states by graph laplacian denoising. *IEEE Transactions on Signal and Information Processing over Networks* **7**:493–508 DOI 10.1109/TSIPN.2021.3100302.

Chehreghani MH. 2016. K-nearest neighbor search and outlier detection via minimax distances. In: *SIAM international conference on data mining*. 405–413.

Daxberger E, Hernández-Lobato JM. 2019. Bayesian variational autoencoders for unsupervised out-of-distribution detection. ArXiv preprint. arXiv:1912.05651 DOI 10.48550/arXiv.1912.05651.

Erfanin SM, Rajasegarar S, Karunasekera S, Leckie C. 2016. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognition* **58**:121–134 DOI 10.1016/j.patcog.2016.03.028.

Ergen T, Hassan Mirza A, Serdar Kozat S. 2017. Unsupervised and semi-supervised anomaly detection with LSTM neural networks. ArXiv preprint. arXiv:1710.09207.

Goodfellow I, Bengio Y, Courville A. 2019. *Deep learning*. Cambridge: MIT Press.

Grathwohl W, Wang K-C, Jacobsen J-H, Duvenaud D, Norouzi M, Swersky K. 2019. Your classifier is secretly an energy based model and you should treat it like one. In: *International conference on learning representations*. 1–23.

Grosnit A, Maraval AM, Tutunov R, Griffiths R-R, Cowen-Rivers AI, Yang L, Zhu L, Lyu W, Chen Z, Wang J, Peters J, Bou-Ammar H. 2022. High-dimensional Bayesian optimisation with variational autoencoders and deep metric learning. ArXiv preprint. arXiv:2106.03609 DOI 10.48550/arXiv.2106.03609.

Grosse K, Manoharan P, Papernot N, Backes M, McDaniel P. 2017. On the (statistical) detection of adversarial examples. ArXiv preprint. arXiv:1702.06280.

Ian G, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. 2014. Generative adversarial nets. In: *Advances in neural information processing systems*. 2672–2680.

Inoue J, Yamagata Y, Chen Y, Poskitt CM, Sun J. 2017. Anomaly detection for a water treatment system using unsupervised machine learning. In: *Data Mining Workshops (ICDMW)*. Piscataway: IEEE, 1058–1065.

Jayasumana S, Hartley R, Salzmann M, Li H, Harandi M. 2013. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In: *CVPR*.

Jayasumana S, Hartley R, Salzmann M, Li H, Harandi M. 2014. Optimizing over radial kernels on compact manifolds. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Piscataway: IEEE, 3802–3809.

Jerónimo Pasadas D, Geirinhas Ramos H, Feng B, Baskaran P, Lopes Ribeiro A. 2020. Defect classification with SVM and wideband excitation in multilayer aluminum plates. *IEEE Transactions on Instrumentation and Measurement* **69(1)**:241–248 DOI 10.1109/TIM.2019.2893009.

Kam Ho T. 1995. Random decision forests. In: *Proceedings of the third international conference on document analysis and recognition*. 278–282.

Kingma DP, Lei Ba J. 2015. Adam: a method for stochastic optimization. ArXiv preprint. arXiv:1412.6980v8.

Li D, Chen D, Shi L, Jin B, Goh J, Ng S-K. 2019. MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. In: *International Conference on Artificial Neural Networks (ICANN) 2019: artificial neural networks and machine learning—ICANN 2019: text and time series*. 703–716.

Li X, Lv J, Yi Z. 2020. Outlier detection using structural scores in a high-dimensional space. *IEEE Transactions on Cybernetics* **50(5)**:2302–2310 DOI 10.1109/TCYB.2018.2876615.

Li Z, Bai X, Hu R, Li X. 2021. Measuring phase-amplitude coupling based on the Jensen–Shannon divergence and correlation matrix. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **29**:1375–1385 DOI 10.1109/TNSRE.2021.3095510.

Lu W, Cheng Y, Xiao C, Chang S, Huang S, Liang B, Huang T. 2017. Unsupervised sequential outlier detection with deep architectures. *IEEE Transactions on Image Processing* **26(9)**:4321–4330 DOI 10.1109/TIP.2017.2713048.

Menon V, Kalyani S. 2019. Structured and unstructured outlier identification for robust PCA: a fast parameter free algorithm. *IEEE Transactions on Signal Processing* **67(9)**:2439–2452 DOI 10.1109/TSP.2019.2905826.

Metzen JH, Genewein T, Fischer V, Bischoff B. 2017. On detecting adversarial perturbations. ArXiv preprint. arXiv:1702.04267.

Olshausen BA, Field DJ. 1997. Sparse coding with an over complete basis set: a strategy employed by V1. *Vision Research* **37**:3311–3325 DOI 10.1016/S0042-6989(97)00169-7.

Parchami M, Bashbaghi S, Granger E, Sayed S. 2017. Using deep autoencoders to learn robust domain-invariant representations for still-to-video face recognition. In: *Advanced Video and Signal Based Surveillance (AVSS), 14th IEEE International Conference on 2017*. Piscataway: IEEE, 1–6.

Peng X, Xu D. 2013. A twin-hypersphere support vector machine classifier and the fast learning algorithm. *Information Science* **221**:12–27 DOI 10.1016/j.ins.2012.09.009.

Popolin Neto M, Paulovich FV. 2021. Explainable matrix-visualization for global and local interpretability of random forest classification ensembles. *IEEE Transactions on Visualization and Computer Graphics* **27(2)**:1427–1437 DOI 10.1109/TVCG.2020.3030354.

Ruff L, Görnitz N, Deecke L, Ahmed Siddiqui S, Binder A, Müller E, Kloft M. 2018. Deep one-class classification. 4390–4399.

Shi Q, Zhang H. 2021. Fault diagnosis of an autonomous vehicle with an improved svm algorithm subject to unbalanced datasets. *IEEE Transactions on Industrial Electronics* **68(7)**:6248–6256 DOI 10.1109/TIE.2020.2994868.

Soleimani H, Miller DJ. 2016. Atd: anomalous topic discovery in high dimensional discrete data. *IEEE Transactions on Knowledge and Data Engineering* **28(9)**:2267–2280 DOI 10.1109/TKDE.2016.2561288.

Song H, Jiang Z, Men A, Yang B. 2017. A hybrid semi-supervised anomaly detection model for high-dimensional data. *Computational Intelligence and Neuroscience* **2017**:8501683.

Wang G, Lu J, Choi K-S, Zhang G. 2020a. A tansfer-based additive LS-SVM classifier for handing missing data. *IEEE Transactions on Cybernetics* **50(2)**:739–752 DOI 10.1109/TCYB.2018.2872800.

Wang H, Pang G, Shen C, Ma C. 2020b. Unsupervised representation learning by predicting random distances. In: *Twenty-ninth international joint conference on artificial intelligence main track*. 2950–2956.

Yu K, Chen H. 2019. Markov boundary-based outlier mining. *IEEE Transactions on Neural Networks and Learning Systems* **30(4)**:1259–1264 DOI 10.1109/TNNLS.2018.2861743.

Yuan Y, Xun G, Ma F, Wang Y, Du N, Jia K, Su L, Zhang A. 2018. Muvan: a multi-view attention network for multivariate temporal data. In: *2018 IEEE International Conference on Data Mining (ICDM)*. Piscataway: IEEE, 717–726.

Zheng C, Zhao C. 2020. Multiclass oblique random forests with dual-incremental learning capacity. *IEEE Transactions on Neural Networks and Learning Systems* **31(12)**:5192–5203 DOI 10.1109/TNNLS.2020.2964737.