frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 08 April 2024
DOI 10.3389/fnins.2024.1362510

OPEN ACCESS

EDITED BY
Anguo Zhang,
University of Macau, China

REVIEWED BY
Omid Memarian Sorkhabi,
University College Dublin, Ireland
Junyi Wu,
Fuzhou University, China

*CORRESPONDENCE
Vladimir Golovko
✉ vladimir.golovko@gmail.com

RECEIVED 28 December 2023
ACCEPTED 28 February 2024
PUBLISHED 08 April 2024

CITATION
Chen C, Golovko V, Kroshchanka A, Mikhno E, Chodyka M and Lichograj P (2024) An analytical approach for unsupervised learning rate estimation using rectified linear units. *Front. Neurosci.* 18:1362510. doi: 10.3389/fnins.2024.1362510

An analytical approach for unsupervised learning rate estimation using rectified linear units

Chaoxiang Chen[1,2,3], Vladimir Golovko[4,5]*, Aliaksandr Kroshchanka[5], Egor Mikhno[5], Marta Chodyka[4] and Piotr Lichograj[4]

[1]School of Information Science and Technology, Zhejiang Shuren University, Hangzhou, China, [2]International Science and Technology Cooperation Base of Zhejiang Province: Remote Sensing Image Processing and Application, Hangzhou, China, [3]Institute of Traditional Chinese Medicine Artificial Intelligence Zhejiang Shuren University, Hangzhou, China, [4]Department of Computer Science, John Paul II University in Biala Podlaska, Biala Podlaska, Poland, [5]Intelligent Information Technologies Department, Brest State Technical University, Brest, Belarus

Unsupervised learning based on restricted Boltzmann machine or autoencoders has become an important research domain in the area of neural networks. In this paper mathematical expressions to adaptive learning step calculation for RBM with ReLU transfer function are proposed. As a result, we can automatically estimate the step size that minimizes the loss function of the neural network and correspondingly update the learning step in every iteration. We give a theoretical justification for the proposed adaptive learning rate approach, which is based on the steepest descent method. The proposed technique for adaptive learning rate estimation is compared with the existing constant step and Adam methods in terms of generalization ability and loss function. We demonstrate that the proposed approach provides better performance.

KEYWORDS

adaptive training step, RBM, deep learning, unsupervised learning, ReLU, activation function, Adam

1 Introduction

During recent years many papers have been devoted to the study of restricted Boltzmann machines (RBM) and more generally to that of deep learning, because it is a breakthrough approach in the field of artificial intelligence (Hinton, 2002, 2010; Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Nair and Hinton, 2010; Krizhevsky et al., 2012; LeCun et al., 2015). Deep learning has been developing very quickly in the last decade. As a result, various successful applications of deep learning have been proposed in speech recognition, computer vision, natural language processing, data visualization, etc. (Bengio et al., 2007, 2013, 2021; Bengio, 2009; Larochelle et al., 2009; Erhan et al., 2010; Golovko et al., 2010; Glorot et al., 2011; Mikolov et al., 2011; Hinton et al., 2012; Madani et al., 2018; Lamb et al., 2022; Verma et al., 2022; Aguilera et al., 2023; Menezes et al., 2023; Chen et al., 2024).

One of the major and important problems in this domain is the selection of suitable hyperparameters values to achieve significant performance of a neural network. Among these parameters, the learning rate is of great importance because it has a significant impact on the training efficiency of the neural network (Golovko, 2003; Cho et al., 2011; Duchi

et al., 2011; Krizhevsky and Hinton, 2012; Zeiler, 2012; Schaul et al., 2013; Kingma and Ba, 2014; Ruder, 2016; Pouyanfar and Chen, 2017; Smith, 2017; Baydin et al., 2018; Takase et al., 2018; Arpit and Bengio, 2019; Vaswani et al., 2019; Pesme et al., 2020; Carvalho et al., 2021; Nakamura et al., 2021; Chen et al., 2022; Defazio et al., 2023; Golovko et al., 2023; Wang et al., 2023). The choice of an appropriate learning rate controls how well the neural network adapts to the problem being solved and achieves a suitable minimum of the loss function. So, for instance, for many applications the learning rate has to be manually and carefully chosen, because depending on this parameter the learning process can be divergent or convergent. Therefore, to avoid these problems, learning step should be defined and modified automatically during neural network learning.

The neural networks community has been concerned with this problem for many years, and currently there are only partial solutions to selecting an appropriate learning rate. This situation gives rise to the question of how we can obtain analytical expressions for learning rate calculation. This question is addressed in the present paper. As a result, an analytical approach to estimate the value of the learning step has been proposed, based on the steepest descent approach. The proposed approach capables to automatically defining and adjusting the learning rate during the training of a neural network.

In our previous work (Golovko et al., 2023), we proposed an approach to estimate the learning rate of a single-layer perceptron with a rectified linear unit activation function (ReLU). The present article focuses on an adaptive learning step (ATS) for RBM with a ReLU. It is the simplest activation function, which is a piecewise linear function consisting of two straight lines. ReLU is not a saturated activation function with unlimited output, unlike other activation functions. It has been noted in existing literature that using a ReLU network generally improves performance (Vaswani et al., 2019; Wang et al., 2023). As stated in the article (Nair and Hinton, 2010) rectified linear units can improve RBM. As well is known a RBM can be applied for deep neural networks learning (LeCun et al., 2015). The conventional approach to RBM learning usually uses constant or empirically varying learning step (Cho et al., 2011). Currently, there are no analytical expressions to estimate the learning rate, which can be automatically defining and adjusting the learning rate during the training of a RBM network. As a rule, there are only empirical and heuristic approaches to set learning rate.

Therefore, in this paper we investigate the calculation of adaptive learning rate for a RBM, which is based on the steepest descent technique (Golovko et al., 2000, 2023; Golovko, 2003). This approach is based on minimizing the loss function to calculate the adaptive learning step. Since derivation an accurate analytical expression for estimating the learning rate using steepest descent approach is a very difficult task, most scientists use the steepest descent method together with the line search approach. However, as we will show in this article, it is possible to derive exact expressions for the RBM learning rate using the ReLU activation function. The adaptive learning rate approach permits to compute the learning step at each time. An advantage of the proposed approach is that we can automatically estimate a specific learning rate value for each batch or each example from the training data set.

Further, we perform stacking ReLU RBM into a deep neural network. As a result, we can train deep neural networks using unsupervised and SGD techniques.

The major contribution of this paper is novel mathematical expressions for adaptive learning rate calculation, if we use RBM with ReLU transfer function. The proposed approach is based on steepest descent technique and allows to estimate the ATS at each iteration of the learning algorithm. We have shown, using a set of experiments, that the proposed adaptive learning rate can improve performance with respect to learning quality and generalization ability.

In the present study we proceed as follows. Section 2 introduces the related work in this area. In Section 3 we consider different representations of RBM. Section 4 deals with learning rules for RBM with ReLU. In section 5 we propose the adaptive learning step calculation for RBM. Section 6 demonstrates the results of experiments, and finally we give our conclusion.

2 Related work

In the following, a brief overview of related works in this area is presented. It is well known that there are the two principal techniques for learning of deep neural networks (DNN): learning with pretraining using a greedy layer wise approach and stochastic gradient descent approach (SGD), including its various modifications. If we do not use pretraining of DNN, then it is necessary to use a rectified linear unit (ReLU) transfer function, because of the vanishing gradient problem (LeCun et al., 2015).

RBM can be used as building blocks for deep neural networks, where every layer of neural network is trained as RBM in an unsupervised manner (Hinton, 2002, 2010; Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Nair and Hinton, 2010). By stacking RBMs in this way, one can obtain a suitable initialization of a deep neural network for further training using a backpropagation algorithm.

For smaller data sets, unsupervised pretraining helps to prevent overfitting (LeCun et al., 2015). As stated in paper (LeCun et al., 2015): "Although at present the supervised training with ReLU is used mainly for deep neural networks learning, we expect unsupervised learning to become far more important in the longer term. Human and animal learning is largely unsupervised: we discover the structure of the world by observing it, not by being told the name of every object." Consequently, unsupervised learning is of great importance. Therefore, we consider in this work the different representations of RBM and study estimation of an adaptive learning rate.

Currently the most methods for learning rate estimation are oriented to the SGD approach (Duchi et al., 2011; Zeiler, 2012; Schaul et al., 2013; Kingma and Ba, 2014; Ruder, 2016; Pouyanfar and Chen, 2017; Smith, 2017; Baydin et al., 2018; Takase et al., 2018; Vaswani et al., 2019; Nakamura et al., 2021; Chen et al., 2022; Defazio et al., 2023; Wang et al., 2023). If the SGD approach is used, then, as a rule, an initial learning rate is selected manually, and further during the learning, the training rate is decreased over time, using different rules. We have not found any works as concerns analytical expressions for the learning rate estimation. There are various approaches to learning rate estimation using different versions of SGD. Let us consider these approaches shortly.

Existing works related to learning rate selection are based mostly on learning rate schedule or line search approach. So, for instance the estimation of adaptive learning rate using line search approach is proposed in Vaswani et al. (2019) and Wang et al. (2023). As mentioned earlier, as a rule, the line search approach is used in

conjunction with the steepest descent technique. However, such an approach is computationally expensive and time consuming. Furthermore, as will be shown in this paper, it is possible to obtain for RBM with ReLU precise expressions for the learning rate instead of using line search. Learning rate scheduling is a very popular approach and is used in various gradient descent optimization algorithms, namely, Adagrad, Adadelta, RMSprop, and Adam. The primary shortcoming associated with learning rate schedules is their dependence on predefined initial learning rate.

So, for instance, the Adagrad method (Duchi et al., 2011) divides the learning rate at each step by the norm of all previous gradients. The other approaches, such as Adadelta and Adam are based on Adagrad and as a result the learning rate decreases during training (Kingma and Ba, 2014; Ruder, 2016). In Pesme et al. (2020), the optimization process of SGD is divided into two stages: transient stage and stationary stage. It should be noted that the learning step is reduced during the stationary phase. In Smith (2017), scheduling learning rate is performed for each iteration. In Baydin et al. (2018), the hypergradient descent approach is proposed in order to find appropriate learning step. In Nakamura et al. (2021), ATS technique is proposed, which is based on a combination of reducing and increasing the learning rate.

As regards analytical learning rate at the pretraining stage, we have not found any works as concerns the learning rate estimation. Substantially, all known approaches are based again not on analytical expressions for calculating the learning rate, but on empirical approaches and the policy of changing the learning step. So, for instance, in Cho et al. (2011) for RBM is proposed an approach to automatically adjust the learning rate by maximizing a local likelihood estimate. However, as a result, the learning rate is chosen based on the previous learning rate and a small constant, that leads again in manual selection of the initial parameters.

In this paper we propose to use steepest descent approach to derive learning rate. Such learning rate can only be obtained for linear and ReLU activation functions. When using the sigmoid activation function, we can only receive approximate expressions for the learning rate using the Taylor series expansion (Golovko et al., 2000; Golovko, 2003). Since this is a very complicated problem, as mentioned before, most of the scientists use the steepest descent method together with the line search approach.

Our previous work (Golovko et al., 2023) reported an adaptive learning rate for a single-layer perceptron with a ReLU activation function. Let us consider the simplest neural network, namely single layer perceptron (SLP). In the case of a single-layer perceptron with ReLU activation function, the expressions for calculating the adaptive learning step was obtained for the first time in the work (Golovko et al., 2023) based on the proof of the following theorems:

Theorem 1: For a single-layer perceptron with a ReLU activation function in the case of online learning, the value of the adaptive learning step is calculated based on the following expression Eq. (1):

$$\alpha(t)=\frac{\sum_{j=1}^{m} r_j(t+1)b_j\left(r_j(t+1)S_j(t)-e_j\right)}{\sum_{j=1}^{m} r_j^2(t+1)b_j^2} \tag{1}$$

$$b_j=r_j(t)\left(y_j-e_j\right)\left(1+\sum_{i=1}^{n}x_i^2(t)\right),$$

where $r_j(t+1)=\begin{cases}\eta_1, e_j(t)\geq 0;\\ r_2, e_j(t)<0.\end{cases}$

Here η_1 and r_2 denotes corresponding slopes of the ReLU function; $e_j(t)$ is desired output for j-th unit; n and m denotes the number of input and output unit, $S_j(t)$, $y_j(t)$ are weighted sum and output of the j-th unit.

It should be noted, that $\eta_1 \neq r_2$ and $0 < r_2 < 1$.

Theorem 2: For a single-layer perceptron with a ReLU activation function in the case of batch learning, the value of the adaptive learning step is calculated based on the following expression Eq. (2):

$$\alpha(t)=\frac{\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(t+1)S_j^k(t)-e_j^k\right)\left(r_j^k(t+1)b_j^k\right)}{\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(t+1)b_j^k\right)^2} \tag{2}$$

$$b_j^k=\sum_{p=1}^{L_1} r_j^p(t)\left(y_j^p-e_j^p\right)\left(1+\sum_{i=1}^{n}x_i^k(t)x_i^p(t)\right),$$

whereL_1 is batch size and $r_j^k(t+1)=\begin{cases}r_1^k, e_j^k(t)\geq 0;\\ r_2^k, e_j^k(t)<0.\end{cases}$

As stated in Golovko et al. (2023), the above expressions Eqs. (1, 2) can significantly increase the learning quality of a single-layer perceptron and achieve an optimal solution to the problem. The proposed approach was generalized to unsupervised pretraining of deep neural network (Golovko et al., 2023), using autoencoder method. The primary goal of the present work is to obtain the analytical expressions to learning rate estimation for restricted Boltzmann machine with ReLU activation function.

3 Restricted Boltzmann machine

In this section we consider different representation of RBM from structure and learning point of view.

Let us consider a conventional restricted Boltzmann machine (Hinton, 2010), which has bipartite structure consisting of two layers: a visible layer containing n units and hidden layer containing m units (Figure 1).

In the RBM structure, each neuron in visible layer is connected to all the units in the hidden layer, using bidirectional weights W. RBM can be used as main building blocks for deep neural networks (Hinton, 2002, 2010; Hinton et al., 2006; Nair and Hinton, 2010). Usually the states of visible and hidden units are defined using a probabilistic version of the sigmoid activation function according to Eqs. (3, 4):

$$p(y_j \mid x)=\frac{1}{1+e^{-S_j}},\quad S_j=\sum_{i=1}^{n} w_{ij}x_i+T_j \tag{3}$$

$$p(x_i \mid x)=\frac{1}{1+e^{-S_i}},\quad S_i=\sum_{j=1}^{m} w_{ij}y_j+T_i \tag{4}$$

It should be noted that the variables at the hidden layer are independent given the state of the visible units, and vice versa as shown in expression Eq. (5):

$$P(x \mid y) = \prod_{i=1}^{n} P(x_i \mid y) \qquad (5)$$

$$P(y \mid x) = \prod_{j=1}^{m} P(y_j \mid x)$$

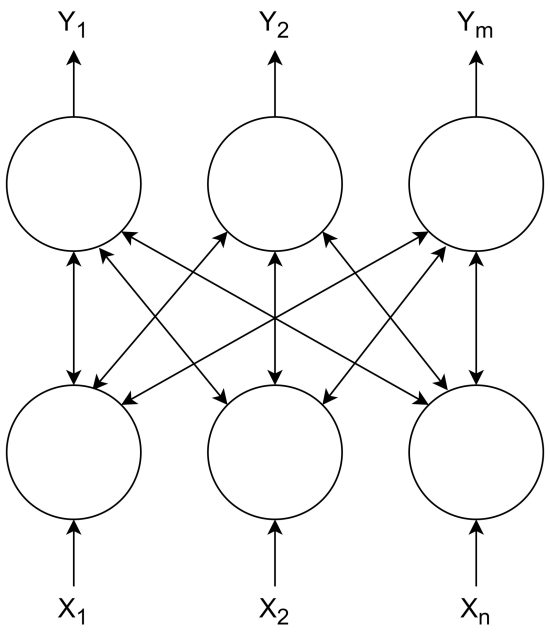

FIGURE 1
Restricted Boltzmann machine.

The hidden units of the RBM can be interpreted as feature detectors which capture the regularities of the input data. The traditional way of getting the training rule is to maximize the function of log-likelihood of the input data distribution P(x). In other words, it is necessary to reproduce the distribution of input data as closely as possible using the states of hidden units. The main properties of conventional RBM are the following: symmetric weights in the hidden and visible layers; Gibbs sampling during the training and stochastic neurons. Next, we will consider a RBM that is characterized only by the first two properties, and the neurons are not stochastic.

Let us consider unfolded representation of the RBM using three layers (visible, hidden and visible; Golovko et al., 2015, 2016) as shown in Figure 2. Such a representation of RBM is equivalent to PCA or autoencoder neural network, where the hidden and last visible layer is, respectively, compression and reconstruction layer.

Let us consider the Gibbs sampling using CD-k. In this case we can represent Gibbs sampling for above structure as shown in Figure 3.

Next, we will consider Gibbs sampling for CD-1. Let x(0) is the input data, that enter at the visible layer at time 0. Then the output of the hidden layer is defined as follows Eqs. (6, 7):

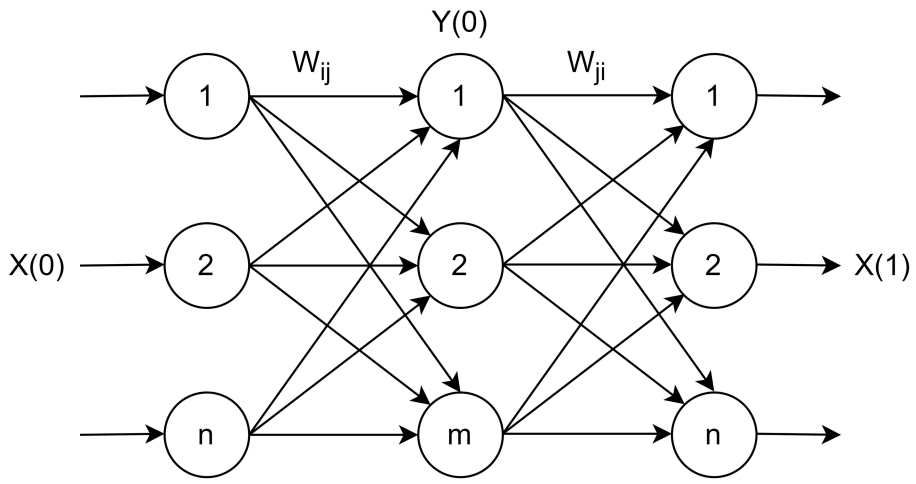

FIGURE 2
Unfolded representation of RBM.

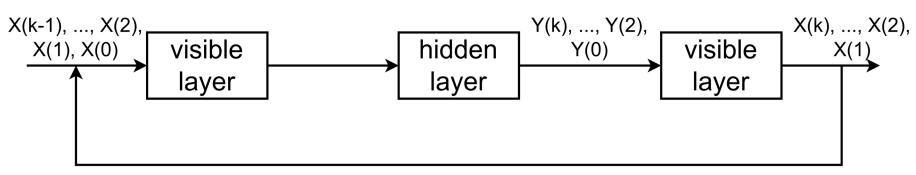

FIGURE 3
Gibbs sampling.

$$y_j(0) = F\left(S_j(0)\right) \tag{6}$$

$$S_j(0) = \sum_i \omega_{ij} x_i(0) + T_j \tag{7}$$

The reconstruction layer reproducts the data from the hidden layer. As a result we can obtain x(1) at time 1 using Eqs. (8, 9):

$$x_i(1) = F\left(S_i(1)\right) \tag{8}$$

$$S_i(1) = \sum_j \omega_{ij} y_j(0) + T_i \tag{9}$$

After this, x(1) enters the visible layer and we can obtain the output of the hidden layer the following way Eqs. (10, 11):

$$y_j(1) = F\left(S_j(1)\right) \tag{10}$$

$$S_j(1) = \sum_i \omega_{ij} x_i(1) + T_j \tag{11}$$

As mentioned before the conventional approach of getting the training rule is to maximize the function of log-likelihood of the input data distribution. In Golovko et al. (2015, 2016), we have proposed an alternative approach in order to obtain RBM learning rule, which is based on the minimization of mean square error (MSE). As stated in Golovko et al. (2016) the primary goal of training RBM is to minimize the reconstruction mean squared error (MSE) in the hidden and visible layers simultaneously. The MSE in the hidden layer is proportional to the difference between the states of the hidden units at the various time steps. Then in case of CD-1 the MSE in the hidden layer is defined as shown in expression Eq. (12):

$$E_h(1) = \frac{1}{2}\sum_{l=1}^{L}\sum_{j=1}^{m}\left(y_j^k(1) - y_j^k(0)\right)^2 \tag{12}$$

Similarly, the MSE in the inverse layer is proportional to the difference between the states of the inverse units at the various time steps Eq. (13):

$$E_v(1) = \frac{1}{2}\sum_{l=1}^{L}\sum_{i=1}^{n}(\left(x_i^k(1) - x_i^k(0)\right)^2 \tag{13}$$

where L is the number of training patterns.

Then the main purpose of the training RBM is to minimize the total mean squared error (MSE), which is defined as the sum of errors Eq. (14):

$$E_s = E_h(1) + E_v(1) \tag{14}$$

The following theorem is proved in Golovko et al. (2016).

Theorem 3: Maximization of the log-likelihood input data distribution P(x) in the space of synaptic weights of the restricted Boltzmann machine is equivalent to special case of minimizing the reconstruction mean squared error in the same space.

As a result, the following training rule was obtained for online learning Eq. (15):

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \alpha\begin{pmatrix}\left(y_j(1) - y_j(0)\right)F'\left(S_j(1)\right)x_i(1)\\ +\left(x_i(1) - x_i(0)\right)F'\left(S_i(1)\right)y_j(0)\end{pmatrix} \tag{15}$$

$$T_i(t+1) = T_i(t) - \alpha\left(x_i(1) - x_i(0)\right)F'\left(S_i(1)\right)$$

$$T_j(t+1) = T_j(t) - \alpha\left(y_j(1) - y_j(0)\right)F'\left(S_j(1)\right)$$

where α is learning rate.

It is easy to show, that if

$$F'\left(S_j(1)\right) = \frac{\partial y_j(1)}{\partial S_j(1)} = F'(S_i(1) = \frac{\partial x_i(1)}{\partial S_i(1)} = 1,$$

then can be obtained the conventional learning rule Eq. (16):

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha\left(x_i(0)y_j(0) - x_i(1)y_j(1)\right),$$

$$T_j(t+1) = T_j(t) + \alpha\left(y_j(0) - y_j(1)\right) \tag{16}$$

$$T_i(t+1) = T_i(t) + \alpha\left(x_i(0) - x_i(1)\right).$$

We have seen in this section, that depending on the loss function can be obtained different learning rules with derivatives and without derivatives of activation function with respect to weighted sum. In further we will use the learning rule with derivatives.

4 Learning of RBM with ReLU

In this section, we consider the definition of ReLU activation function and RBM learning rule. As noted earlier, we consider RBM with deterministic neurons and for learning we will use the expressions given in the previous section. First of all, let us define the ReLU activation function by the following way.

Definition: The ReLU activation function for j-th unit can be presented by the following way Eq. (17):

$$y_j(t) = F\left(S_j(t)\right) = r_j(t)S_j(t) \tag{17}$$

where $r_j(t)$ is defined by the following way Eq. (18):

$$r_j(t) = \begin{cases}\eta_1, S_j(t) \geq 0;\\ r_2, S_j(t) < 0.\end{cases} \tag{18}$$

Here η_1 and r_2 denote corresponding slopes of the ReLU function; $\eta_1 \neq r_2$; $0 \leq r_2 < 1$. Usually $\eta_1 = 1$ is used.

The above definition of the activation function allows the use of any slope of straight lines and generalizes the conventional definition of ReLU and leaky ReLU activation functions.

Then we can obtain the following derivatives Eq. (19):

$$F'\left(S_j(1)\right)=\frac{\partial y_j(1)}{\partial S_j(1)}=r_j(1)\ and$$
$$F'\left(S_i(1)\right)=\frac{\partial x_i(1)}{\partial S_i(1)}=r_i(1) \tag{19}$$

Using the previous results Eq. (15), we can write the following equations for online learning Eq. (20):

$$\omega_{ij}(t+1)=\omega_{ij}(t)-\alpha\begin{pmatrix}\left(y_j(1)-y_j(0)\right)r_j(1)x_i(1)\\+\left(x_i(1)-x_i(0)\right)r_i(1)y_j(0)\end{pmatrix} \tag{20}$$

$$T_i(t+1)=T_i(t)-\alpha\left(x_i(1)-x_i(0)\right)r_i(1)$$

$$T_j(t+1)=T_j(t)-\alpha\left(y_j(1)-y_j(0)\right)r_j(1)$$

If we apply the batch learning and batch size is L_1,

$$\omega_{ij}(t+1)=\omega_{ij}(t)-\alpha\sum_{p=1}^{L_1}\begin{pmatrix}\left(y_j^p(1)-y_j^p(0)\right)r_j^p(1)x_i^p(1)\\+\left(x_i^p(1)-x_i^p(0)\right)r_i^p(1)y_j^p(0)\end{pmatrix} \tag{21}$$

$$T_i(t+1)=T_i(t)-\alpha\sum_{p=1}^{L_1}\left(x_i^p(1)-x_i^p(0)\right)r_i^p(1)$$

$$T_j(t+1)=T_j(t)-\alpha\sum_{p=1}^{L_1}\left(y_j^p(1)-y_j^p(0)\right)r_j^p(1)$$

Thus, in this section, we have derived learning rules for RBM with ReLU activation function. Further we will use given above expressions Eq. (21) for RBM learning.

5 Materials and methods

In this section we address adaptive learning rate estimation for RBM with ReLU activation function. Since the RBM network has symmetric weights in the hidden and visible layers, we should derive the optimal training step for the two layers.

The learning step is called adaptive, which is chosen at each stage of the algorithm in such a way in order to minimize the total mean squared error (Golovko et al., 2000; Golovko, 2003; Golovko et al., 2023). We will use the steepest descent approach in order to obtain the expression for adaptive learning rate. Accordingly, to steepest descent approach, the learning step α is selected so as to minimize the mean square error of the new parameters Eq. (22):

$$\alpha(t)=\min E_s\left(y_j^k(1,t+1),x_i^k(1,t+1)\right) \tag{22}$$

where $y_j^k(1,t+1),x_i^k(1,t+1)$ are the outputs of the hidden and visible layer at the next time $t+1$ after updating the RBM trainable parameters.

As a result, at each step of learning algorithm we should choose the value of learning rate in such a way that, when modifying weights and thresholds to guarantee a minimum of the mean squared error for each batch or each example from the training data set.

Theorem 4: For an RBM network with a ReLU activation function in the case of batch learning, the value of the adaptive learning step, that minimizes the mean squared error for each batch is calculated based on the following expression:

$$\alpha(t)=\frac{\sum_{k=1}^{L_1}\sum_{j=1}^{m}c_j^k+\sum_{k=1}^{L_1}\sum_{i=1}^{n}c_i^k}{\begin{matrix}\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(1,t+1)\right)^2\left(b_j^k\right)^2\\+\sum_{k=1}^{L_1}\sum_{i=1}^{n}\left(r_i^k(1,t+1)\right)^2\left(b_i^k\right)^2\end{matrix}} \tag{23}$$

where the corresponding terms are determined according to the expressions Eqs. (24–32):

$$c_j^k=\left(r_j^k(1,t+1)S_j^k(1)-y_j^k(0)\right)r_j^k(1,t+1)b_j^k \tag{24}$$

$$c_i^k=\left(r_i^k(1,t+1)S_i^k(1)-x_i^k(0)\right)r_i^k(1,t+1)b_i^k \tag{25}$$

$$b_j^k=f_j^k+z_j^k \tag{26}$$

$$f_j^k=\sum_{p=1}^{L_1}r_j^p(1)\left(y_j^p(1)-y_j^p(0)\right)\left(1+\sum_{i=1}^{n}x_i^k(1)x_i^p(1)\right) \tag{27}$$

$$z_j^k=\sum_{p=1}^{L_1}y_j^p(0)\sum_{i=1}^{n}x_i^k(1)\left(x_i^p(1)-x_i^p(0)\right)r_i^p(1) \tag{28}$$

$$b_i^k=f_i^k+z_i^k, \tag{29}$$

$$f_i^k=\sum_{p=1}^{L_1}r_i^p(1)\left(x_i^p(1)-x_i^p(0)\right)\left(1+\sum_{j=1}^{m}y_j^k(0)y_j^p(0)\right) \tag{30}$$

$$z_i^k=\sum_{p=1}^{L_1}x_i^p(1)\sum_{j=1}^{m}y_j^k(0)\left(y_j^p(1)-y_j^p(0)\right)r_j^p(1) \tag{31}$$

$$r_j^k(1,t+1)=\begin{cases}r_1, y_j^k(0)\geq 0;\\ r_2, y_j^k(0)<0,\end{cases}\ r_i^k(1,t+1)=\begin{cases}r_1, x_i^k(0)\geq 0;\\ r_2, x_i^k(0)<0.\end{cases} \tag{32}$$

Proof: We should find adaptive learning rate by minimizing the following loss function Eq. (33):

$$E_s=\frac{1}{2}\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(y_j^k(1,t+1)-y_j^k(0)\right)^2+\frac{1}{2}\sum_{k=1}^{L_1}\sum_{i=1}^{n}\left(x_i^k(1,t+1)-x_i^k(0)\right)^2 \tag{33}$$

The output of the hidden and visible layer at the next time $t+1$ after updating trainable parameters can be defined as according to the expressions Eq. (34):

$$x_i^k(1,t+1)=r_i^k(1,t+1)S_i^k(1,t+1),\\ y_j^k(1,t+1)=r_j^k(1,t+1)S_j^k(1,t+1) \tag{34}$$

Let us consider at the beginning the weighted sum of the hidden layer at the next time $t+1$

$$S_j^k(1,t+1)=\sum_{i=1}^{n}\omega_{ij}(t+1)x_i^k(1)+T_j(t+1) \tag{35}$$

Substituting corresponding expression for weights and threshold updating from Eq. (21) in Eq. (35) we can obtain Eq. (36)

$$S_j^k(1,t+1)=S_j^k(1)-\alpha b_j^k \tag{36}$$

where b_j is defined using Eq. (37)

$$b_j^k=\sum_{p=1}^{L_1}r_j^p(1)\left(y_j^p(1)-y_j^p(0)\right)\left(1+\sum_{i=1}^{n}x_i^k(1)x_i^p(1)\right)\\ +\sum_{p=1}^{L_1}y_j^p(0)\sum_{i=1}^{n}x_i^k(1)\left(x_i^p(1)-x_i^p(0)\right)r_i^p(1) \tag{37}$$

We will use a similar approach for the visible layer. Then the weighted sum of the visible layer can be defined as follows:

$$S_i^k(1,t+1)=\sum_{j=1}^{m}\omega_{ij}(t+1)y_j^k(0)+T_i(t+1) \tag{38}$$

Substituting corresponding expression from Eq. (21) in Eq. (38) we can write obtain Eq. (39)

$$S_i^k(1,t+1)=S_i^k(1)-\alpha b_i^k \tag{39}$$

Where b_j is defined using Eq. (40)

$$b_i^k=\sum_{p=1}^{L_1}r_i^p(1)\left(x_i^p(1)-x_i^p(0)\right)\left(1+\sum_{j=1}^{m}y_j^k(0)y_i^p(0)\right)\\ +\sum_{p=1}^{L_1}x_i^p(1)\sum_{j=1}^{m}y_j^k(0)\left(y_j^p(1)-y_j^p(0)\right)r_j^p(1) \tag{40}$$

As a result, we can obtain the final expressions regarding output of the hidden and visible layer Eq. (41)

$$y_j^k(1,t+1)=r_j^k(1,t+1)\left(S_j^k(1)-\alpha b_j^k\right)\\ x_i^k(1,t+1)=r_i^k(1,t+1)\left(S_i^k(1)-\alpha b_i^k\right). \tag{41}$$

Differentiating the loss function E_s with respect to α we can obtain Eq. (42)

$$\frac{dE_s}{d\alpha}=\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(1,t+1)S_j^k(1)-\alpha r_j^k(1,t+1)b_j^k-y_j^k(0)\right)\\ \left(-r_j(1,t+1)b_j\right)+\sum_{k=1}^{L_1}\sum_{i=1}^{n}\left(r_i^k(t+1)S_i^k(1)\right.\\ \left.-\alpha r_i^k(1,t+1)b_i^k-x_i^k(0)\right)\\ \left(-r_i^k(1,t+1)b_i^k\right)=0 \tag{42}$$

As a result, we can obtain the following final expression Eq. (43):

$$\alpha(t)=\frac{\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(1,t+1)S_j^k(1)-y_j^k(0)\right)r_j^k(1,t+1)b_j^k}{+\sum_{k=1}^{L_1}\sum_{i=1}^{n}\left(r_i^k(1,t+1)S_i^k(1)-x_i^k(0)\right)r_i^k(1,t+1)b_i^k}\\ \frac{}{\sum_{k=1}^{L_1}\sum_{j=1}^{m}\left(r_j^k(1,t+1)\right)^2\left(b_j^k\right)^2}\\ +\sum_{k=1}^{L_1}\sum_{i=1}^{n}\left(r_i^k(1,t+1)\right)^2\left(b_i^k\right)^2 \tag{43}$$

Since in accordance with Eq. (44)

$$\frac{d^2E_s}{d^2\alpha}>0 \tag{44}$$

we have found the minimum of the cost function. Thus the theorem is proved.

As follows from the proven theorem, the adaptive learning rate minimizes the mean squared error of the network under updating weights and thresholds.

The major difficulty arises in the computing of $r_j^k(1,t+1), r_i^k(1,t+1)$, because it is desired parameters of ReLU transfer function. Since the desired outputs of the hidden and visible layer correspondingly $y_j^k(0)$ and $x_i^k(0)$ then we can write Eq. (45)

$$r_j^k(1,t+1)=\begin{cases}r_1, y_j^k(0)\geq 0;\\ r_2, y_j^k(0)<0,\end{cases} r_i^k(1,t+1)=\begin{cases}r_1, x_i^k(0)\geq 0;\\ r_2, x_i^k(0)<0.\end{cases} \tag{45}$$

Theorem 5: For RBM network with a ReLU activation function in the case of online learning, the value of the adaptive learning step, that minimizes the mean squared error for each pattern is defined as follows Eq. (46):

$$\alpha(t)=\frac{\sum_{j=1}^{m}c_j+\sum_{i=1}^{n}c_i}{\sum_{j=1}^{m}r_j^2(1,t+1)b_j^2+\sum_{i=1}^{n}r_i^2(1,t+1)b_i^2} \tag{46}$$

where the corresponding terms are calculated based on Eqs. (47–55)

$$c_j=\left(r_j(1,t+1)S_j(1)-y_j(0)\right)r_j(1,t+1)b_j \tag{47}$$

$$c_i=\left(r_i(1,t+1)S_i(1)-x_i(0)\right)r_i(1,t+1)b_i \tag{48}$$

$$b_j=f_j+z_j \tag{49}$$

$$f_j=r_j(1)\left(y_j(1)-y_j(0)\right)\left(1+\sum_{i=1}^{n}x_i^2(1)\right) \tag{50}$$

$$z_j=y_j(0)\sum_{i=1}^{n}x_i(1)\left(x_i(1)-x_i(0)\right)r_i(1) \tag{51}$$

$$b_i=f_i+z_i \tag{52}$$

$$f_i=r_i(1)\left(x_i(1)-x_i(0)\right)\left(1+\sum_{j=1}^{m}y_j^2(0)\right) \tag{53}$$

$$z_i = x_i(1)\sum_{j=1}^{m} y_j(0)\left(y_j(1)-y_j(0)\right)r_j(1) \quad (54)$$

$$r_j(1,t+1)=\begin{cases}\eta_1, y_j(0)\geq 0;\\ r_2, y_j(0)<0,\end{cases} r_i(1,t+1)=\begin{cases}\eta_1, x_i(0)\geq 0;\\ r_2, x_i(0)<0.\end{cases} \quad (55)$$

This theorem is proved by the same approach.

It should be noted that the proposed expressions for calculating the learning step are valid when $r_2 \neq 0$. If $r_2 \neq 0$, then in accordance with RBM learning rule Eq. (21), the training is performed only in the area where weighted sum is greater than 0, since the gradient of this function is 0, if weighted sum less than 0. In that case, we can simplify the expressions for learning rate.

Corollary: For RBM network with a ReLU activation function and $\eta_1 = 1, r_2 = 0$ in the case of online learning, the value of the adaptive learning step is calculated based on the following expression Eqs. (56–58):

$$\alpha(t)=\frac{\sum_{j=1}^{m}\left(y_j(1)-y_j(0)\right)b_j+\sum_{i=1}^{n}\left(x_i(1)-x_i(0)\right)b_i}{\sum_{j=1}^{m}b_j^2+\sum_{i=1}^{n}b_i^2} \quad (56)$$

Where

$$b_j=\left(y_j(1)-y_j(0)\right)-\sum_{i=1}^{n}x_i(1)\left(y_j(0)x_i(0)-x_i(1)y_j(1)\right) \quad (57)$$

$$b_i=\left(x_i(1)-x_i(0)\right)-\sum_{j=1}^{n}y_j(0)\left(y_j(0)x_i(0)-x_i(1)y_j(1)\right) \quad (58)$$

In a similar way, we can obtain an expression for the adaptive step calculation when using batch learning. The proposed expressions allow to estimate the learning rate after presenting every batch or pattern to the neural network and based on the minimization of loss function. Adaptive training step approach permits to choose automatically step for every batch or pattern from training data set. The performance of proposed approach is discussed in the next section.

6 Experiments

This section summarizes numerical results obtained by the application of adaptive and constant learning rate. In order to evaluate the performance of the proposed approach we will conduct various experiments using RBM and deep neural network. In all experiments, we will use batch learning with adaptive rate Eq. (23). In that case the weights and thresholds of the network will be modified based on rule Eq. (21) presented in this paper. For experiments, we will use both an artificial and the MNIST dataset. The primary aim of this section is to compare learning of neural network with and without proposed training approach with adaptive learning rate. The experiments are divided into 2 groups. The first experiments focuses on the RBM network and the second on deep multilayer neural network.

6.1 RBM results

Let us consider the use of an adaptive learning step for a RBM network. To evaluate the effectiveness of adaptive learning rate we will use two datasets.

6.1.1 Artificial dataset

The artificial data x lie on a one-dimensional manifold (a helical loop) embedded in three dimensions (Scholz et al., 2008) and were generated from a uniformly distributed factor t in the range [0.05, 0.95]:

$$\begin{cases} x_1 = \sin(\pi t) + \mu, \\ x_2 = \cos(\pi t) + \mu, \\ x_3 = t + \mu. \end{cases}$$

where μ – Gaussian noise with mean 0 and standard deviation 0.05.

The primary goal of the experiment is to study the performance of ATS for data compression and reconstruction. Then the RBM will consist of 3 visible and 1 hidden unit. The training dataset consists of 1,000 samples. The size of the test patterns is also 1,000. The batch size equal 8 and the parameters of ReLU function are the following: $\eta_1 = 1, r_2 = 0.01$. We trained the RBM network using only clean data and tested using noisy data. The evolution of reconstruction error vs. epoch of RBM learning is provided in Table 1. As can be seen from the table the adaptive learning rate has obvious excellence compared to constant steps. The plots of the reconstruction accuracy vs. epoch for learning and testing using the best constant and adaptive rate are presented in Figures 4, 5. It should be noted here that testing is performed after each learning epoch. As follows from the presented figures, the adaptive learning rate has the evident advantage compared to the fixed learning rate, namely, the best performance in terms of learning quality and generalization ability.

6.1.2 MNIST dataset

In this section we will use the MNIST dataset, which contains 60,000 hand-written digit images for training, and 10,000 images for testing. Data in MNIST are grayscale images with size 28 × 28. Before training the images are normalized to be zero-mean.

TABLE 1 Evolution of the reconstruction error (MSE) for artificial data.

Number of epochs	α = 3e − 1	α = 3e − 2	α = 3e − 3	α = 3e − 4	α = 3e − 5	ATS
10	0.2223	0.0974	0.1394	0.1393	0.1862	0.1001
20	0.2240	0.0975	0.0982	0.1392	0.1653	0.0903
30	0.2241	0.0982	0.0983	0.1392	0.1537	0.0902
40	0.2230	0.0972	0.0982	0.1392	0.1474	0.0737
50	0.2231	0.0972	0.0982	0.1391	0.1422	0.0753

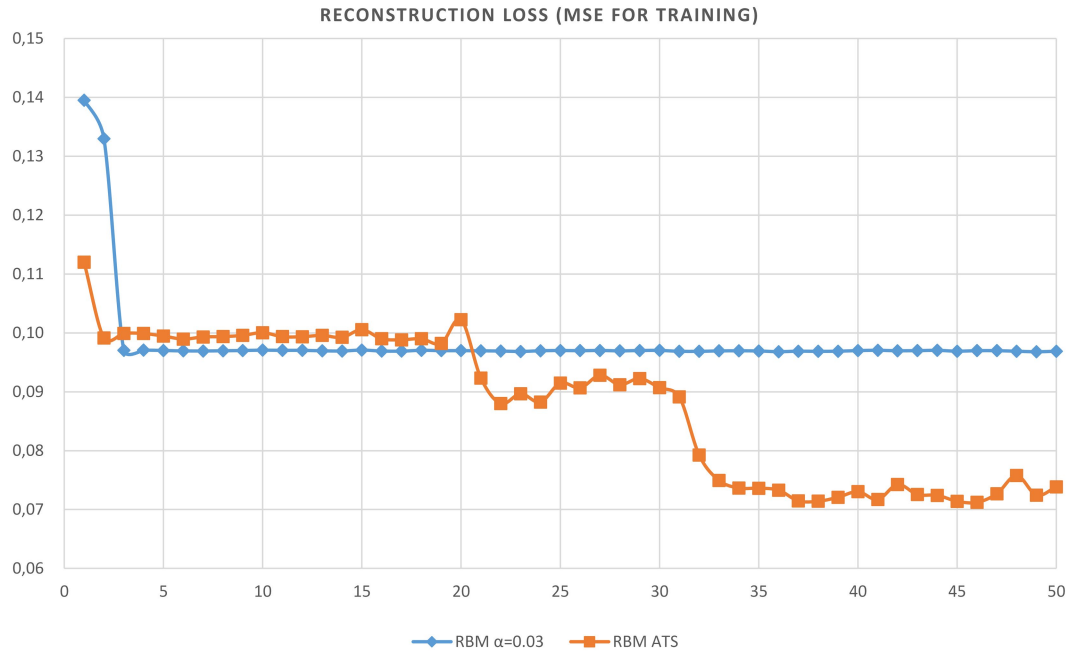

FIGURE 4
Plot of reconstruction accuracy versus epoch for learning using adaptive and constant learning rates.

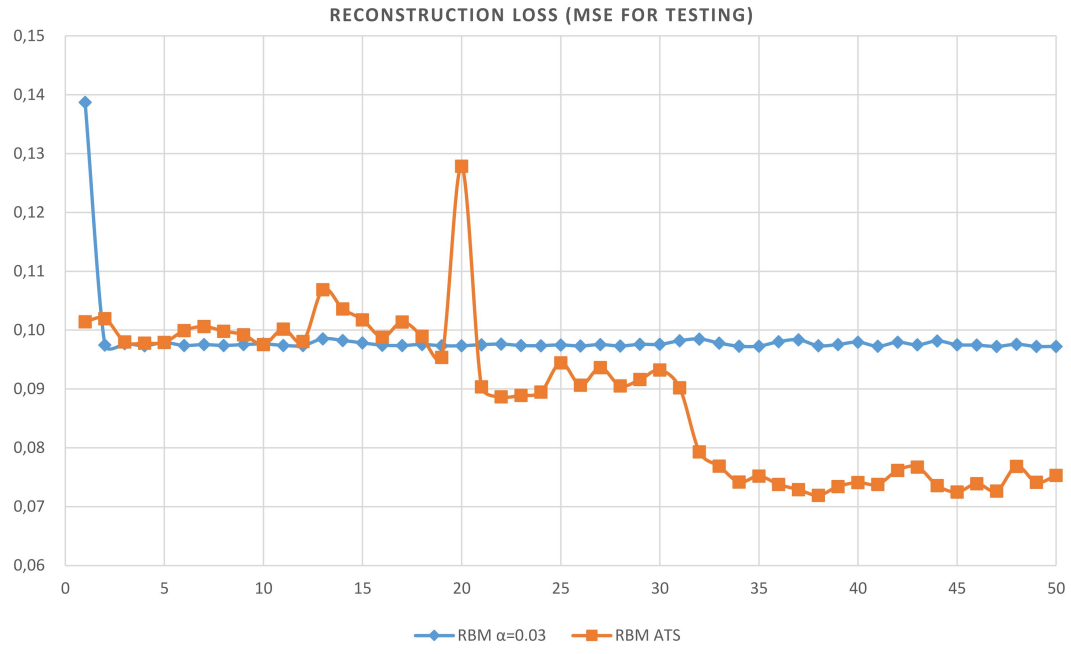

FIGURE 5
Plot of reconstruction accuracy versus epoch for testing using adaptive and constant learning rates.

Let us model the RBM network. This simulation is used to illustrate the compression and reconstruction properties of restricted Boltzmann machine. Let us model the RBM network which consist of 784 neurons of visible and 128 units of hidden layers. The main goal of such modeling is to compress and reconstruct MNIST data. The parameters of experiments are shown in Table 2. We used original images from MNIST dataset and before representation to RBM only centering is performed.

The evolution of reconstruction square error Eq. (13) is shown in Table 3. Here div. Denotes divergence of learning. The analysis of the data in this table indicates that learning with a constant rate is unstable. For instance, if $\alpha = 1e - 4$, the neural network cannot be trained. As can be seen only training with a constant learning rate ($3e - 7$) leads to a positive result.

Hence, the learning algorithms with constant training step can diverge if the learning parameters are not chosen appropriately, as

TABLE 2 Parameters of experiments.

Number of pretraining epoch	Batch size	η_1	η_2
8	128	1	0.01

TABLE 3 Evolution of the reconstruction error (MSE) for MNIST.

Number of epochs	$\alpha = 1e-4$	$\alpha = 3e-5$	$\alpha = 3e-6$	$\alpha = 3e-7$	ATS
1	0,126	0.095	0.073	0.073787208	0.070819163
2	0,178	0.111	0.074	0.073663836	0.070752306
3	0,213	0.125	0.075	0.073577446	0.070733909
4	div.	0.138	0.077	0.073518835	0.070713023
5	div.	0.154	0.081	0.073481718	0.070700173
6	div.	0.171	0.087	0.073461681	0.070700861
7	div.	0.199	0.098	0.073455503	0.070687373
8	div.	0.232	0.115	0.073460822	0.070678658

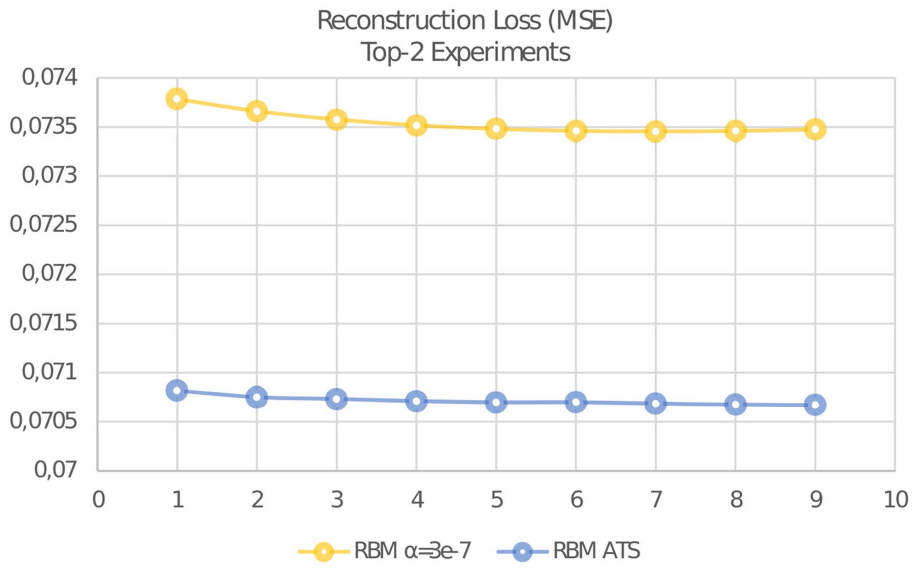

FIGURE 6
Evolution of reconstruction error MSE for adaptive and constant learning rates.

shown in Table 3. Therefore, we should select the constant learning rate very carefully.

Also it should be remarked, that learning with ATS have shown the result after first epoch better than with constant step at any epoch. After 8 epochs have obtained the best result with reconstruction error of 0.070678658. The best result using constant step is 0.073455503. This result was obtained after 8 epochs. As can be seen, the adaptive learning rate has a significant advantage in comparison with the constant learning stage. The evolution of reconstruction error is presented in Figure 6.

6.2 Deep multilayer perceptron

Let us consider the analysis of the proposed approach for a deep multilayer neural network using the MNIST dataset. We have used for experiments deep perceptron with ReLU activation function which has the following structure: 784-1600-1600-800-800-10. The parameters of experiments are shown in Table 4. The results of our experiments are shown in Table 5. Pretraining is performed using only 1–3 epochs.

The evolution of mean squared error for different approaches is presented in Figure 7. Finally, we have the following experimental results, which are shown in Table 5. The smallest test error without using ATS and pretraining is 0.002531. If we use ATS the smallest test error is 0.002299. The smallest test error for pretraining with constant step is 0.002392. As in the previous case, the table shows that learning with a constant rate can be unstable. As a result, a learning algorithm with a constant learning step may be divergent. Tables 6–8 show the predictive performance of different learning approaches for MNIST classification. As can be seen, in general the ATS approach outperformed the learning technique with fixed learning rate. So, for

TABLE 4 MNIST Classification Experiment Parameters.

Number of pretraining epochs	Number of fine-tuning epochs	Batch size	r_1	r_2	Initial α at the finetuning stage
3	50	128	1	0.01	3e − 4

TABLE 5 Testing a deep multilayer perceptron pertaining.

	α = 1e − 4	α = 3e − 5	α = 3e − 6	α = 3e − 7	ADAM	ATS
MSE	div	div	0.0024235	0.0024619	0.0024729	0.0022999
Precision macro	0.0098	0.0098	0.986	0.98562	0.98539	0.98656
Accuracy	0.0098	0.0098	0.986	0.9857	0.9854	0.9866

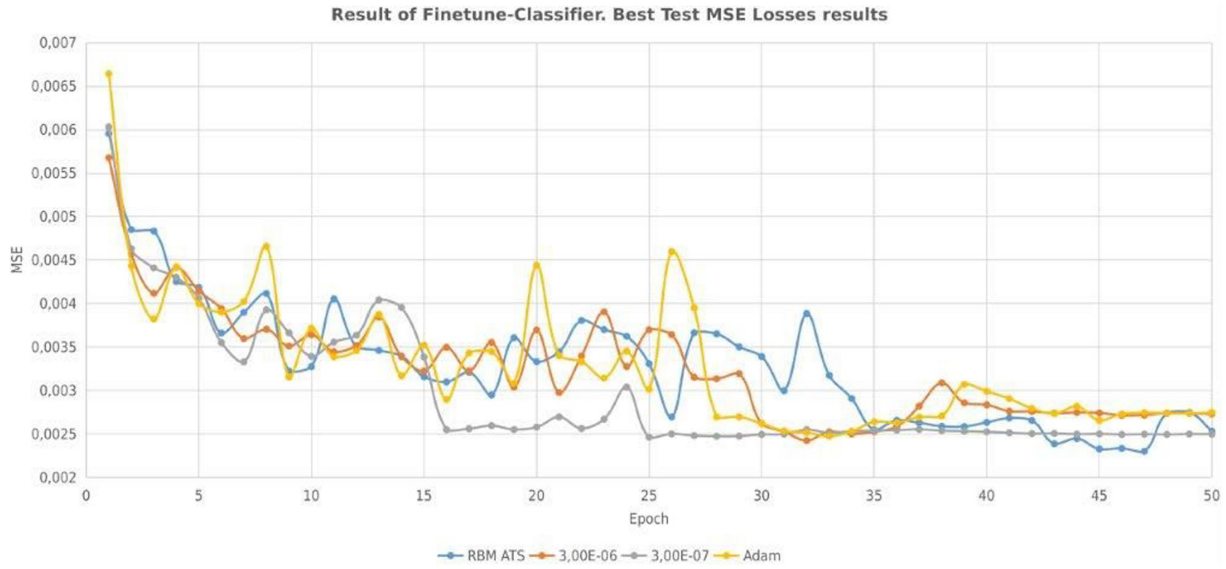

FIGURE 7
Evolution of MSE for adaptive and constant learning rates.

instance, the number of correct predictions using the adaptive rate (Table 6) is 6 and 9 more, respectively, compared to models with a constant step (Tables 7, 8). As can be seen the use of ATS permits to reduce the test error and correspondingly improve the generalization ability.

7 Conclusion and discussion

The learning of neural networks is a tricky task, which highly depends on suitable hyperparameters selection to achieve significant performance of a neural network. The choice of an appropriate learning rate is of great importance because it has a significant impact on the training efficiency. Depending on this parameter the learning process can be divergent or convergent.

We have not found any works as concerns exact analytical expressions for the learning rate estimation, based on the steepest descent technique. Such precise analytical expressions can only be obtained for linear and ReLU activation functions. When using the sigmoid activation function, we can only receive approximate expressions for the learning rate using the Taylor series expansion. Since this is a very complicated problem, most of the scientists use the steepest descent method together with the line search approach.

Our previous work (Golovko et al., 2023) reported an adaptive learning rate for a single-layer perceptron with a ReLU activation function. In this work, we extended this idea to obtain the learning rate for the RBM network. As a result, novel analytical expressions for learning step estimation have been proposed in this paper. The proposed approach for ATS estimation is based on minimization the mean squared error for each batch or each sample. The presented expressions are applied for restricted Boltzmann machine learning with ReLU activation function. We consider quasi-conventional RBM, namely we use symmetric weights in the hidden and visible layers, Gibbs sampling and deterministic units. We first demonstrate the proposed approach for ATS calculation is more effective and more efficient for RBM learning than the conventional RBM algorithm. Second, we show that such kind of RBM can be used for deep neural network pretraining using greedy layer wise algorithm. As a result, we can reach better generalization ability.

The main advantages of the proposed approach are the following: it is based on precise mathematical expressions obtained by minimizing the mean squared error for each batch

TABLE 6 Confusion matrix for MNIST classification using pretraining with ATS.

		Predicted values									
		0	1	2	3	4	5	6	7	8	9
Actual values	0	974	0	0	0	0	0	1	2	2	1
	1	0	1,130	0	1	0	1	1	2	0	0
	2	2	0	1,015	2	2	0	0	5	5	1
	3	0	0	1	998	0	3	0	4	3	1
	4	1	0	1	1	966	0	2	1	0	10
	5	2	0	0	5	0	877	2	1	4	1
	6	4	2	0	0	1	4	947	0	0	0
	7	1	2	6	0	0	0	0	1,013	2	4
	8	1	0	1	3	1	0	0	5	960	3
	9	1	2	1	4	7	3	0	4	1	986

TABLE 7 Confusion matrix for MNIST classification using pretraining with $\alpha = 3e - 6$.

		Predicted values									
		0	1	2	3	4	5	6	7	8	9
Actual values	0	973	1	1	0	0	0	3	1	1	0
	1	0	1,126	1	2	0	2	2	1	1	0
	2	1	1	1,018	2	1	0	1	7	1	0
	3	0	0	3	999	0	2	0	2	2	2
	4	1	2	1	0	969	0	2	0	0	7
	5	2	0	0	5	1	878	2	1	1	2
	6	5	2	0	0	3	2	945	0	1	0
	7	1	2	7	1	1	0	0	1,010	3	3
	8	4	0	3	7	2	1	1	4	950	2
	9	3	2	0	0	6	3	0	3	0	992

TABLE 8 Confusion matrix for MNIST classification using pretraining with $\alpha = 3e\text{-}7$.

		Predicted values									
		0	1	2	3	4	5	6	7	8	9
Actual values	0	974	1	0	1	0	1	1	1	1	0
	1	0	1,126	3	0	0	2	3	0	1	0
	2	0	1	1,021	2	1	0	1	4	2	0
	3	0	0	4	999	0	1	0	1	0	5
	4	1	1	3	0	964	0	3	3	0	7
	5	2	0	0	3	1	881	1	1	2	1
	6	6	2	0	1	4	4	940	0	1	0
	7	2	1	5	3	0	0	0	1,015	2	0
	8	3	0	2	6	1	3	1	2	952	4
	9	4	2	0	3	6	4	0	3	2	985

or each pattern; it is capable of automatically defining and adjusting the learning rate during neural network training; it guarantees convergence to well-performing local minima. The disadvantage of the presented approach is higher computational complexity compared to constant step.

This work opens the way toward the following future research: define the conditions where such an learning rate can guarantee convergence to best-performing local minima and to study how this approach can be extended to train a multilayer neural network without pretraining using RBM.

Data availability statement

Publicly available datasets were analyzed in this study. This data can be found at: https://yann.lecun.com/exdb/mnist/.

Author contributions

CC: Data curation, Funding acquisition, Investigation, Writing – original draft. VG: Conceptualization, Formal analysis, Supervision, Writing – original draft. AK: Methodology, Project administration, Visualization, Writing – review & editing. EM: Software, Validation, Writing – review & editing. MC: Funding acquisition, Investigation, Writing – review & editing. PL: Methodology, Resources, Software, Writing – review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This research was partially supported by the Ministry of Science and Technology of the People's Republic of China (grant number G2022016010L) and Belarusian Republican Foundation for Fundamental Research (grant Ф22КИ-046).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Aguilera, A., Olmos, P., Artés-Rodríguez, A., and Pérez-Cruz, F. (2023). Regularizing transformers with deep probabilistic layers. *Neural Netw.* 161, 565–574. doi: 10.1016/j.neunet.2023.01.032

Arpit, D., and Bengio, Y. (2019). The benefits of overparameterization at initialization in deep ReLU networks. arXiv [Preprint]. arXiv:1901.03611.

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. (2018). Online learning rate adaptation with hypergradient descent. In *Proceedings of International Conference on Learning Representations*.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundat Trends Machine Learn* 2, 1–127. doi: 10.1561/2200000006

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning a review and new perspectives. In: *Institute of Electrical and Electronics Engineers transactions on pattern analysis and machine intelligence*; 35, 1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). "Greedy layer-wise training of deep networks" in *Advances in neural information processing systems*. eds. J. C. Scholkopf, T. Platt and S. Hoffman, MIT Press, Cambridge vol. 11, 153–160.

Bengio, Y., Lecun, Y., and Hinton, G. (2021). Deep learning for AI. *Commun. ACM* 64, 58–65. doi: 10.1145/3448250

Carvalho, P., Lourencco, N., and Machado, P. (2021). Evolving learning rate optimizers for deep neural networks. arXiv [Preprint]. arXiv:2103.12623.

Chen, B., Wang, H., and Ba, C. (2022). Differentiable self-adaptive learning rate. ArXiv [Preprint]. arXiv:2210.10290.

Chen, K., Weng, Y., Hosseini, A., Dening, T., Zuo, G., and Zhang, Y. (2024). A comparative study of GNN and MLP based machine learning for the diagnosis of Alzheimer's disease involving data synthesis. *Neural Netw.* 169, 442–452. doi: 10.1016/j.neunet.2023.10.040

Cho, K., Raiko, T., and Ilin, A. (2011). Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In: *Proceedings of the 28th International Conference on Machine Learning*, *ICML*, Bellevue, Washington, USA.

Defazio, A., Cutkosky, A., Mehta, H., and Mishchenko, K. (2023). When, why and how much? Adaptive learning rate scheduling by refinement. arXiv [Preprint]. arXiv:2310.07831.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* 12, 257–269. doi: 10.5555/1953048.2021068

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* 11, 625–660.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier networks. In *Proceedings of the 14th international conference on artificial intelligence and statistics*. JMLR W&CP; 15, 315–323.

Golovko, V. (2003). "From neural networks to intelligent systems: selected aspects of training, application and evolution" in ed. Marco Gori. *Limitations and future trends in neural computation* (Amsterdam: IOS Press), 219–243.

Golovko, V., Komar, M., and Sachenko, A. (2010). Principles of neural network artificial immune system design to detect attacks on computers. In *Proceedings of the international Conference on Modern Problems of Radio Engineering (TSET)*, p.237.

Golovko, V., Kroshchanka, A., and Treadwell, D. (2016). The nature of unsupervised learning in deep neural networks: a new understanding and novel approach. *Optic Memory Neural Netw* 25, 127–141. doi: 10.3103/S1060992X16030073

Golovko, V., Kroshchanka, A., Turchenko, V., Jankowski, S., and Treadwell, D. (2015). A new technique for restricted Boltzmann machine learning. In *Proceedings of the 8th IEEE international conference IDAACS*, pp.182–186.

Golovko, V., Mikhno, E., Kroschanka, A., Chodyka, M., and Lichograj, P. (2023). Adaptive learning rate for unsupervised learning of deep neural networks. *International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6.

Golovko, V., Savitsky, Y., Laopoulos, T., Sachenko, A., and Grandinetti, L. (2000). Technique of learning rate estimation for efficient training of MLP. In *Proceedings of the IEEE-INNS-ENNS international joint conference on neural networks, IJCNN*. Neural computing: New challenges and perspectives for the new millennium; pp. 323–328.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14, 1771–1800. doi: 10.1162/089976602760128018

Hinton, G. E. (2010). *A practical guide to training restricted Boltzmann machines*, Toronto: Machine Learning Group, University of Toronto.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A. R., Jaitly, N., et al. (2012). Deep neural networks for acoustic Modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* 29, 82–97. doi: 10.1109/MSP.2012.2205597

Hinton, G., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 1527–1554. doi: 10.1162/neco.2006.18.7.1527

Hinton, G., and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science* 313, 504–507. doi: 10.1126/science.1127647

Kingma, D. P., and Ba, J. (2014). *Adam: A method for stochastic optimization, Computer Science,* arXiv preprint.

Krizhevsky, I. S., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Krizhevsky, A., Sutskever, L., and Hinton, G. (2012). Image net classification with deep convolutional neural networks. In proc. *Adv. Neural Inf. Proces. Syst.* 25, 1090–1098.

Lamb, A., Verma, V., Kawaguchi, K., Matyasko, A., Khosla, S., Kannala, J., et al. (2022). Interpolated adversarial training: achieving robust neural networks without sacrificing too much accuracy. *Neural Netw.* 154, 218–233. doi: 10.1016/j.neunet.2022.07.012

Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *J. Mach. Learn. Res.* 1, 1–40. doi: 10.1145/1577069.1577070

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521, 436–444. doi: 10.1038/nature14539

Madani, K., Kachurka, V., Sabourin, C., Amarger, V., Golovko, V., and Rossi, L. (2018). A human-like visual-attention-based artificial vision system for wildland firefighting assistance. *Appl. Intell.* 48, 2157–2179. doi: 10.1007/s10489-017-1053-6

Menezes, A., de Moura, G., Alves, C., and de Carvalho, A. C. P. L. F. (2023). Continual object detection: a review of definitions, strategies, and challenges. *Neural Netw.* 161, 476–493. doi: 10.1016/j.neunet.2023.01.041

Mikolov, T., Deoras, A., Povey, D., Burget, L., and Cernocky, J. (2011). "Strategies for training large scale neural network language models" in *Automatic Speech Recognition and Understanding*, 195–201.

Nair, V., and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp.807–814.

Nakamura, K., Derbel, B., Won, K., and Hong, B. (2021). Learning-rate annealing methods for deep neural networks. *Electronics* 10:2029:2029. doi: 10.3390/electronics10162029

Pesme, S., Dieuleveut, A., and Flammarion, N. (2020). On convergence-diagnostic based step sizes for stochastic gradient descent. In: *Proceedings of the international conference on machine learning, ICML*, 119, pp. 7641–7651.

Pouyanfar, S., and Chen, S. C. (2017). T-LRA: trend-based learning rate annealing for deep neural networks. In *Proceedings of the 2017 IEEE third international conference on multimedia big data (BigMM)*, Laguna Hills, CA, USA; pp. 50–57.

Ruder, S. (2016). An overview of gradient descent optimization algorithms, Available at: https://arxiv.org/abs/1609.04747.

Schaul, T., Zhang, S., and LeCun, Y. (2013). No more pesky learning rates. In *Proceedings of the international conference on machine learning (ICML-2013)*, Atlanta, GA, USA; pp. 343–351.

Scholz, M., Fraunholz, M., and Selbig, J. (2008). *Nonlinear principal component analysis: Neural network models and applications, in principal manifolds for data visualization and dimension reduction*, Springer Berlin Heidelberg, 44–67.

Smith, L. N. (2017). Cyclical learning rates for training neural networks. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

Takase, T., Oyama, S., and Kurihara, M. (2018). Effective neural network training with adaptive learning rate based on training loss. *Neural Netw.* 101, 68–78. doi: 10.1016/j.neunet.2018.01.016

Vaswani, S., Mishkin, A., Laradji, I., Schmidt, M., Gidel, G., and Lacoste-Julien, S. (2019). Painless stochastic gradient: interpolation, line-search, and convergence rates. *Adv. Neural Inf. Proces. Syst.*

Verma, V., Kawaguchi, K., Lamb, A., Kannala, J., Solin, A., Bengio, Y., et al. (2022). Interpolation consistency training for semi-supervised learning. *Neural Netw.* 145, 90–106. doi: 10.1016/j.neunet.2021.10.008

Wang, Z.-J., Gao, H.-B., Wang, X.-H., Zhao, S.-Y., Li, H., and Zhang, X.-Q. (2023). Adaptive learning rate optimization algorithms with dynamic bound based on Barzilai-Borwein method. *Inform. Sci.* 634, 42–54. doi: 10.1016/j.ins.2023.03.050

Zeiler, M. D. (2012). Adadelta: An adaptive learning method. ArXiv [Preprint]. arXiv:1212.5701.