Speedup of Network Training Process by Eliminating the Overshoots of Outputs

Di Zhou(✉), Yuxin Zhao, Chang Liu, and Yanlong Liu

College of Automation, Harbin Engineering University,
Harbin 150001, HLJ, People's Republic of China
{zhoudi128, zhaoyuxin, liuchang407,
yanlong_liu}@hrbeu.edu.cn

Abstract. The overshoots between the expected and actual outputs while training network will slow down the training speed and affect the training accuracy. In this paper, an improved training method for eliminating overshoots is proposed on the basis of traditional network training algorithms and a suggestion of eliminating overshoot is given. Gradient descent is regarded as the training criterion in traditional methods which neglects the side effects caused by overshoots. The overshoot definition (OD) is combined with gradient descent. According to the overshoot suggestion, local linearization and weighted mean methods are used to adjust the parameters of network. Based on the new training strategy, a numerical experiment is conducted to verify the proposed algorithm. The results show that the proposed algorithm eliminates overshoots effectively and improves the training performance of the network greatly.

Keywords: Output overshoots · Local linearization · Weighted mean
Artificial neural network

1 Introduction

Artificial neural networks (ANNs) have developed rapidly and have been used wildly in many fields, such as speech recognition, computer vision, games and classification of skin cancer [1–4]. So far, the studies of ANNs can be divided into three aspects. The first aspect is the topology of network. The numbers of nodes and layers play a significant role in applications of networks. ANNs have developed multiple branch types, such as convolution neural network, recurrent neural network, extreme learning machine [5–7]. However, there is still no substantial instruction on the topology of network. The second aspect is to combine the neural network with other optimization algorithms. Since lots of optimization algorithms exist, this aspect is mainly used for special applications. In [8, 9], ANNs are combined with fuzzy and genetic theory to identify nonlinear system and a short-term load forecasting of natural gas. Network training algorithms are the third aspect and most popular aspect.

Training algorithms play a decisive role in the accuracy of ANNs. In general, a large number of historical data have to be used while training network. Therefore, how to handle big data efficiently and extract features accurately are the key to training network. In [10], Rumelhart proposed learning representations by back-propagating errors.

L. Iliadis et al. (Eds.): AIAI 2018, IFIP AICT 519, pp. 462–470, 2018.
https://doi.org/10.1007/978-3-319-92007-8_39

BP algorithm has good convergence property, however, it may lead to the problem of gradient vanishing. For the disadvantages of BP algorithm, ReLU is proposed to replace the sigmoid activation function [11]. In [12], dropout of nodes is proposed to prevent ANNs from overfitting. In addition, the parallel and online sequential learning algorithms are introduced to quicken the training process [13, 14].

Almost all existing algorithms focus on gradient while few studies take the overshoots of outputs into consideration. An eliminating overshoot method (EOM) is proposed to improve BP algorithm. An overshoot definition is given on the basis of traditional BP (TBP) algorithm. Meanwhile, local linearization and weighted mean algorithms are adopted to train ANNs. This paper is organized as follows. In Sect. 2, we briefly describe the overshoot definition and its mathematical method. Section 3 introduces the numerical simulation. Finally, the conclusion and the future work are summarized in Sect. 4.

2 The Overshoot Definition and Mathematical Method

In this study, the OD combines the outputs with gradient decent. The OD is given as all outputs must be on the same side of the excepted output with the output at the first iteration while training network. The purpose of training network is to adjust the parameters on the basis of a given set of samples to minimize the errors between the actual and expected outputs. Not only gradients but also outputs have to be considered while training networks. A cost function is defined to characterize the errors. At present, there are many kinds of cost functions. The Euclidean metric is used as the cost function in this study, which is defined as follows:

$$E = \frac{1}{2} \cdot (O - D)^T \cdot (O - D) \tag{1}$$

O is the actual output vector, D is the expected response output. T denotes transpose operation. E is the total instantaneous error energy.

The weight and threshold matrixes of the i^{th} layer are defined as follows:

$$W_i = \begin{bmatrix} w_{i,1,1} & w_{i,1,2} & \cdots & w_{i,1,N_{i+1}} \\ w_{i,2,1} & w_{i,2,2} & \cdots & w_{i,2,N_{i+1}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{i,1,1} & w_{i,1,1} & \cdots & w_{i,1,1} \end{bmatrix} \tag{2}$$

$$B^i = \begin{bmatrix} b_1^i \\ b_2^i \\ \vdots \\ b_{N_i}^i \end{bmatrix} \tag{3}$$

$w_{i,j,k}$ denotes the weight value between the j^{th} node of the i^{th} layer and the k^{th} node of the $(i+1)^{th}$ layer. b_i^j denotes the threshold value of the j^{th} node in the i^{th} layer. N_i denotes the number of the nodes in the i^{th} layer.

The output of the i^{th} layer is defined as follows:

$$O^i = \begin{bmatrix} o_1^i \\ o_2^i \\ \vdots \\ o_{N_i}^i \end{bmatrix} \tag{4}$$

According to the Eqs. (2)–(4), the output of the $(i+1)^{th}$ layer can be expressed as follows:

$$O^{i+1} = F(W_i, B^i, O^i) \tag{5}$$

F is the action function.

The correction ΔW_i and ΔB_i are defined as follows:

$$\Delta W_i = -\eta \cdot \frac{dE}{dW_i} = -\eta \cdot \begin{bmatrix} \frac{\partial E}{\partial w_{i,1,1}} & \frac{\partial E}{\partial w_{i,1,2}} & \cdots & \frac{\partial E}{\partial w_{i,1,N_{i+1}}} \\ \frac{\partial E}{\partial w_{i,2,1}} & \frac{\partial E}{\partial w_{i,2,2}} & \cdots & \frac{\partial E}{\partial w_{i,2,N_{i+1}}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{i,N_i,1}} & \frac{\partial E}{\partial w_{i,N_i,2}} & \cdots & \frac{\partial E}{\partial w_{i,N_i,N_{i+1}}} \end{bmatrix} \tag{6}$$

$$\Delta B_i = -\eta \cdot \frac{dE}{dB^i} = -\eta \cdot \begin{bmatrix} \frac{\partial E}{\partial b_1^i} \\ \frac{\partial E}{\partial b_2^i} \\ \vdots \\ \frac{\partial E}{\partial b_{N_i}^i} \end{bmatrix} \tag{7}$$

In Eq. (7), $\frac{dE}{dB_i}$ is the jacobian matrix of E with respect to the threshold vector B^i denoted as J_i^B. The jacobian matrix can be calculated layer by layer.

$$J_i^B = \frac{dE}{dB^i} = \frac{dE}{dO^L} \cdot \frac{dO^L}{dO^{L-1}} \cdots \frac{dO^{i+1}}{dO^i} \frac{dO^i}{dB^i}, \tag{8}$$

Where $i = 1, 2, \cdots, L$.

In Eq. (8), $\frac{dE}{dO^L}$ denotes the jacobian matrix of E with respect to the actual output O^L. L denotes the number of all layers in network.

$$\frac{dE}{dO^L} = \begin{bmatrix} \frac{\partial E}{\partial O_1^L} & \frac{\partial E}{\partial O_2^L} & \cdots & \frac{\partial E}{\partial O_{N_L}^L} \end{bmatrix} = \begin{bmatrix} O_1^L - d_1^L & O_2^L - d_2^L & \cdots & O_{N_L}^L - d_{N_L}^L \end{bmatrix}. \tag{9}$$

d_i^L denotes the expected response.

$\frac{dO^{i+1}}{dO^i}$ denotes the jacobian matrix of the output of the $(i+1)^{th}$ layer O^{i+1} with respect to the output of the i^{th} layer O^i.

$$\frac{dO^{i+1}}{dO^i} = \begin{bmatrix} \frac{\partial o_1^{i+1}}{\partial o_1^i} & \frac{\partial o_1^{i+1}}{\partial o_2^i} & \cdots & \frac{\partial o_1^{i+1}}{\partial o_{L_i}^i} \\ \frac{\partial o_2^{i+1}}{\partial o_1^i} & \frac{\partial o_2^{i+1}}{\partial o_2^i} & \cdots & \frac{\partial o_2^{i+1}}{\partial o_{L_i}^i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial o_{L_{i+1}}^{i+1}}{\partial o_1^i} & \frac{\partial o_{L_{i+1}}^{i+1}}{\partial o_2^i} & \cdots & \frac{\partial o_{L_{i+1}}^{i+1}}{\partial o_{L_i}^i} \end{bmatrix} \tag{10}$$

$\frac{dO^i}{dB^i}$ denotes the jacobian matrix of O^i with respect to B^i.

$$\frac{dO^i}{dB^i} = \begin{bmatrix} \frac{\partial O_1^i}{\partial b_1^i} & \frac{\partial O_1^i}{\partial b_2^i} & \cdots & \frac{\partial O_1^i}{\partial b_{N_i}^i} \\ \frac{\partial O_2^i}{\partial b_1^i} & \frac{\partial O_2^i}{\partial b_2^i} & \cdots & \frac{\partial O_2^i}{\partial b_{N_i}^i} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial O_{N_i}^i}{\partial b_1^i} & \frac{\partial O_{N_i}^i}{\partial b_2^i} & \cdots & \frac{\partial O_{N_i}^i}{\partial b_{N_i}^i} \end{bmatrix} \tag{11}$$

According to the Eqs. (7)–(11), the initial correction ΔB^i can be obtained. The expression of the initial correction ΔW^i can be derived with the same method above. In this study, the sigmoid function is chosen as the action function. The output locates in the interval (0,1). It can be obtained from the Eq. (8) that the correction is determined by the errors of outputs directly.

Assuming that the overshoot arises at the n^{th} iteration, it indicates the corrections exceed the range. Accordingly, the OD should be adopted to eliminate the overshoots for achieving the optimal solution as soon as possible. Since the traditional line search algorithm is time consuming, local linearization and weighted mean methods are used to adjust the parameters to eliminate overshoots in this paper. The local linearization method is based on the relationship between cost function and network parameters. When overshoots occur, the actual outputs of the latest two iterations locate on both sides of the excepted output, which are close to the expected output. In this situation, the error changes little, which is in line with the local linearization condition. The network parameters are regarded as the independent variables, while the cost function is the corresponding variable. Linear equations are established by the latest two iterations, which are defined as follows:

$$\begin{cases} E - E_n = \Delta B_n^i (\overset{\wedge}{B_n^i} - B_n^i) \\ E - E_{n-1} = \Delta B_{n-1}^i (\overset{\wedge}{B_n^i} - B_{n-1}^i) \end{cases} \tag{12}$$

ΔB_n^i is the gradient of E with respect to B^i at the n^{th} iteration. $\hat{B}_n^i$ is the updated network parameters. E_n is the total error at the n^{th} iteration.

The updated network parameters can be obtained by solving the equations when the gradients are nonzero. According to Eq. (12), the updated parameters can be divided into two cases. For the first case, if the cost function is a convex function between B_n^i and B_{n-1}^i, the updated parameters should locate in the interval $(\min(B_{n-1}^i, B_n^i), \max(B_{n-1}^i, B_n^i))$. Then the overshoot of the updated parameters is judged according to the OD. If overshoots are eliminated, the updated parameters are accepted and continue the next iteration. Otherwise, the above step has to be circulated until the overshoots are eliminated. For the second case, the cost function is a non-convex function between B_n^i and B_{n-1}^i. The updated parameters may not locate in the interval $(\min(B_{n-1}^i, B_n^i), \max(B_{n-1}^i, B_n^i))$. The weighted mean method is adopted to deal with this situation. The specific process is defined as follows:

$$\hat{B}_n^i = \frac{|\Delta B_n^i|}{|\Delta B_n^i| + |\Delta B_{n-1}^i|} \cdot B_{n-1}^i + \frac{|\Delta B_{n-1}^i|}{|\Delta B_n^i| + |\Delta B_{n-1}^i|} \tag{13}$$

Similarly to the first case, the updated parameters are accepted if the updated parameters satisfy the overshoot definition, or this step will continue to circulate until overshoots are eliminated.

3 Numerical Simulation and Result Analysis

In this section, a numerical simulation is conducted to verify the effectiveness and practicability of the EOM algorithm. A three layers feed-forward ANN is designed. The sigmoid function is chosen as the action function. The numbers of neurons in input, hidden and output layer are 22, 30 and 58. The input and output samples are selected from the solutions of the downwelling atmosphere monochromatic radiative transfer model equation (MonoRTM). Before training the network, all samples are normalized.

In order to avoid the denominator is 0, the network parameter is accepted when the absolute values of its gradients in the latest two iterations satisfy the overshoot definition and are both below 0.000001. This operation will also speed up the training process.

Similarly to the TBP methods, the OD is also introduced to train the network layer by layer. However, contrary to the direction the error propagates, the OD propagates forwardly. The parameters of the next layer are adjusted on the basis of no overshoot on the previous layers. The TBP algorithm is simulated to compare with the proposed method. The structure of network and initial parameters of the two methods are set to the same. The maximal number of iterations and the acceptable error are set to 50 and 0.000001.

The total instant error of all samples at each iteration is recorded and shown in Fig. 1. The blue and red curves represent the total error of the TBP and the EOM method respectively.

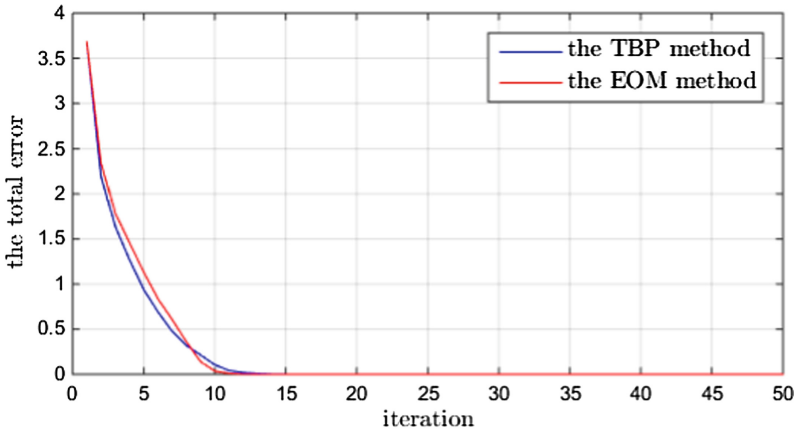

Fig. 1. The total errors of outputs of the EOM and BP methods. The red and blue lines represent the outputs of EOM and BP methods, respectively. (Color figure online)

As can be seen from the figure, the errors of the two methods almost overlap in the initial stage. This is mainly because no overshoot occurs at this stage. The network parameters meet the OD at each iteration which implies that they are not corrected again.

However, the error of the EOM method is reduced at a slower speed than the TBP method in the middle stage. The experimental data shows that errors of the most sample are still large, but overshoots appear at some sample points. In order to satisfy the OD, the parameters leading to overshoots are corrected again, resulting in that the error cannot reduce in the direction of the steepest descent. Therefore, the blue curve is lower than the red curve in the figure. However, the gradient of the blue curve slows down gradually while the red curve almost shows a linear decline trend.

In the final stage, the errors of the samples are small. Overshoots in many sample points lead to a slow speed of training network with traditional methods. However, the EOM method takes great advantage of the good suppression of overshoots and minimizes the error quickly. As shown in Fig. 1, it takes 14 iterations to achieve the optimal solution for the TBP method, however, only 11 for the EOM method.

An output sample point is selected randomly to study the elimination of overshoot detailedly. The actual outputs of the two methods are recorded and shown in Fig. 2. The red and blue curves are respective the actual output of the EOM and TBP method. The green curve is the excepted output.

It can be manifested from Fig. 2, no overshoot appears in the initial stage. It indicates the parameters satisfy the OD. The blue and red curves overlap each other which is consistent with the result in Fig. 1. The correctness of the results in Fig. 1 can be confirmed in this stage.

The blue curve implies the overshoot appears when training network with traditional BP method in the middle stage. In order to meet the OD, the parameters have to be corrected again by the local linearization and weighted mean methods. The red curve depicts the elimination of overshoots of outputs while training neural network. In

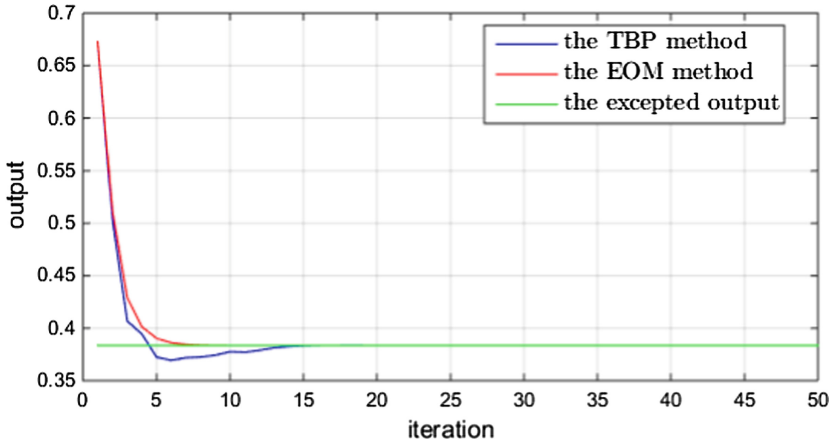

Fig. 2. The outputs of single sample of the EOM and BP methods. The red and blue lines represent the outputs of EOM and BP methods, respectively. The green line is the excepted output (Color figure online)

addition, the EOM method takes 8 iterations to approximate the excepted output which is half of that of the TBP method. Furthermore, the output curve of the EOM method is smoother than that of TBP method.

In order to illustrate the effectiveness of the proposed algorithm to eliminate overshoot intuitively, the training trajectories of 58 output samples of the two methods are shown in Figs. 3 and 4.

The blue points in Fig. 3 and green points in Fig. 4 represent the trajectories of the TBP and EOM method respectively. The red points in both figures are the excepted outputs. Overshoots appear at 90 percent of samples in Fig. 3. And overshoots are

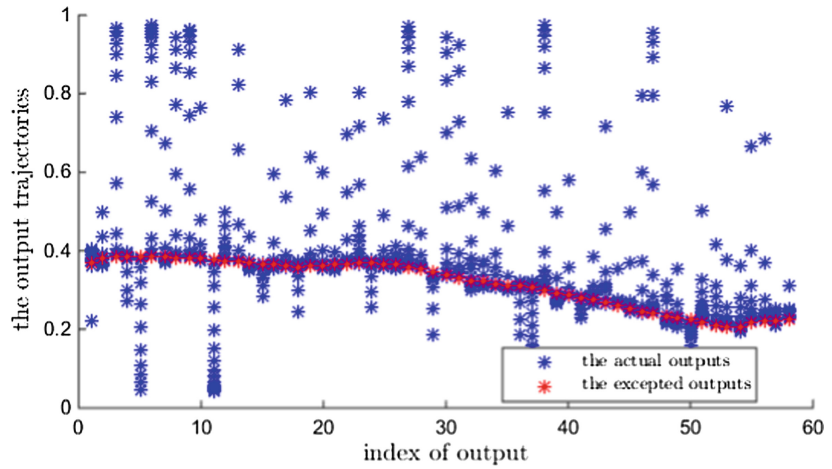

Fig. 3. The output trajectories of the TBP method. (Color figure online)

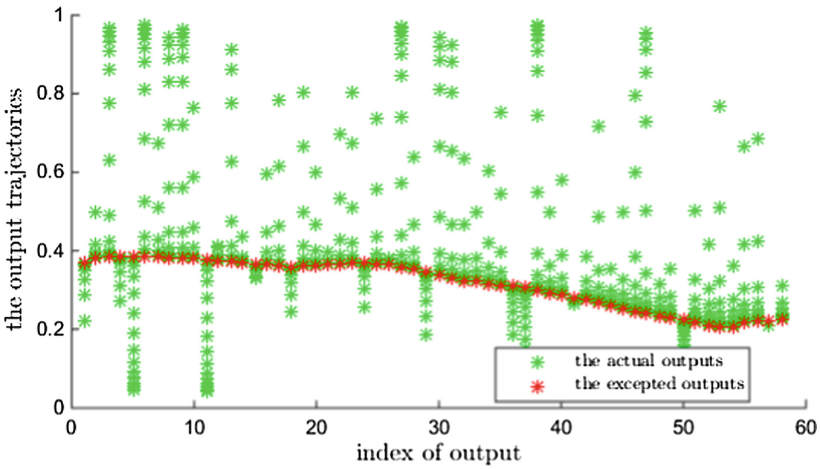

Fig. 4. The output trajectories of the EOM method. (Color figure online)

obvious especially when the initial error is around 0.15. However, no overshoot appears in Fig. 4 which is consistent with the results in Fig. 2. No overshoot for all samples indicates the proposed method is universal. And the OD is universal for other neural network model.

4 Conclusion

In this paper, an improved training method and the OD are proposed for eliminating the overshoots of outputs. The outputs and gradients of the network are taken into consider. The training process of neural network is derived in matrix form and the specific expressions are given. Local linearization and weighted mean methods are introduced to optimize training algorithm. In the final stage of training network, the proposed method eliminates all overshoots perfectly. The advantage of eliminating overshoot makes it possible to reach the optimal earlier than the traditional method. The simulation demonstrates effectiveness of the EOM method and its universality for all samples. Therefore, the EOM method is superior to the traditional method. In addition, a drawback of the EOM method is that it trains network at a slow speed in the middle stage. It is also the direction of our future work.

Acknowledgements. This paper is funded by the International Exchange Program of Harbin Engineering University for Innovation-oriented Talents Cultivation, National Natural Science Foundation of China (Grant Nos. 41676088) and the National Key Research and Development Project of China (2017YFC1404100).

References

1. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Sig. Process. Mag. **29**(6), 82–97 (2012)
2. Gould, S.: DARWIN: a framework for machine learning and computer vision research and development. J. Mach. Learn. Res. **13**, 3533–3537 (2012)
3. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
4. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542** (7639), 115–118 (2017)
5. Mahendran, A., Vedaldi, A.: Visualizing deep convolutional neural networks using natural pre-images. Int. J. Comput. Vision **120**(3), 233–255 (2016)
6. Weninger, F., Bergmann, J., Schuller, B.: Introducing CURRENNT: The munich opensource CUDA recurrent neural network toolkit. J. Mach. Learn. Res. **16**(1), 547–551 (2015)
7. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1), 489–501 (2006)
8. Lin, Y.Y., Chang, J.Y., Lin, C.T.: Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. IEEE Trans. Neural Netw. Learn. Syst. **24**(2), 310–321 (2013)
9. Yu, F., Xu, X.: A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network. Appl. Energy **134**, 102–113 (2014)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by backpropagating errors. Nature **323**(6088), 533 (1986)
11. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for LVCSR using rectified linear units and dropout. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8609–8613. IEEE (2013)
12. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
13. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
14. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)