UDC 004.032.26
DOI: 10.15587/2706-5448.2020.217613

Akanova A.,
Kaldarova M.

IMPACT OF THE COMPILATION METHOD ON DETERMINING THE ACCURACY OF THE ERROR LOSS IN NEURAL NETWORK LEARNING

In the field of NLP (Natural Language Processing) research, the use of a neural network has become important. The neural network is widely used in the semantic analysis of texts in different languages. In connection with the actualization of the processing of big data in the Kazakh language, a neural network was built for deep learning. In this study, the object is the learning process of a deep neural network, which evaluates the algorithm for constructing an LDA model. One of the most problematic places is determining the correct arguments, which, when compiling the model, will give an estimate of the algorithm's performance. During the research, the compile () method from the Keras modular library was used, the main arguments of which are the loss function, optimizers, and metrics. The neural network is implemented in the Python programming language. The main arguments of the neural network deep learning compiler for evaluating the LDA model is the selection of arguments to obtain the correct evaluation of the algorithm of the constructed model using deep learning of the neural network. A corpus of text in the Kazakh language with no more than 8000 words is presented as learning data. Using the above methods, an experiment was carried out on the selection of arguments for the model compiler when learning a text corpus in the Kazakh language. As a result, the optimizer – SGD, the loss function – binary_crossentropy, and the estimation metric – 'cosine_proximity' were chosen as the optimal arguments, which, as a result of learning, showed a tendency to 0 loss (errors)=0.1984, and cosine_proximity (learning accuracy)=0.2239, which is considered acceptable learning measures. The results indicate the correct choice of compilation arguments. These arguments can be applied when conducting deep learning of a neural network, where the sample data is a pair of «topic and keywords».

Keywords: *assessment metric, learning quality, optimization algorithms, entropy error, neural network.*

Received date: 12.08.2020
Accepted date: 22.09.2020
Published date: 31.12.2020

1. Introduction

The main action in deep learning is to adjust the weights to reduce error using a series of learning examples, which in turn boils down to finding the correlation between the input and output layers. If no correlation is present, then the error will never reach 0 [1]. Usually, for learning, a neural network is created, taking into account the parameters of the available data being processed. The main trick of deep learning is using the estimate to adjust the weights to reduce losses. This adjustment is carried out by an optimizer that implements the so-called back propagation algorithm: the central deep learning algorithm [2]. The main learning parameters are usually: the optimizer, the learning coefficient, the sample size, the loss function, the metric for evaluating the correctness of the mask construction, the number of learning epochs.

Therefore, it is relevant to address the following issues:

– studying the Keras library;
– study of the application of the loss function in deep learning of a neural network;
– research of optimization algorithms;
– research of quality indicators.

All of the above parameters will be implemented in the Keras library, and it is not necessary to perform the computational process manually. But a combination of different parameters can show results that will be far from real indicators.

Thus, *the object of research* is the process of learning a deep neural network, which evaluates the algorithm for constructing an LDA model. *The aim of research* is to select the optimal parameters to determine the accuracy of the loss of errors in deep learning of a neural network.

2. Methods of research

In deep learning of a neural network, the complexity of the model is optimized along with accuracy. Complexity is understood as the structural complexity of the model – this is the number of model parameters, taking into account their domain of definition. An alternative to this definition is the statistical complexity of the model, that is, the output of the minimum amount of information that is required to convey information about the model and about the sample.

The optimizer is one of three parameters used in compiling deep learning neural networks. Optimizers or otherwise optimization algorithms are used when learning a neural network, which serve to select weights in such a way as to minimize the error on the learning dataset. Keras implements such basic optimization algorithms as the gradient descent method, the stochastic gradient descent method – SGD, Adagrad, Adadelta, Adam (Adaptive Moment Estimation), etc.

An optimizer is a value that shows the effect of a particular feature on the distribution of a target variable. The SGD optimizer was chosen to optimize the learning process. The argument in this optimizer is learning_rate and the initial learning level should be float>=0. With SGD, the learning rate depends on specific parameters, which are adapted to the frequent parameter updates during learning. The more updates the parameter receives, the lower the learning rate. In other words, the stochastic gradient descent method (SGD) [3] effectively redistributes the learning step for each individual parameter, while taking into account all past gradients for this parameter.

Learning rate refers to the relationship between the learning and test data. For example, validation_split=0.25 means that 75 % of the data will be used to learn the model and the remaining 25 % will be used to test the model.

The sample size or batch_size determines the number of learning samples that are processed per iteration of the gradient descent algorithm.

The loss function or entropy is a measure of the error generated by the system. One of the most popular loss functions used in neural networks is cross-entropy, however, categorical, sparse, or binary cross-entropy is more common in working with NLP. Entropy works directly with the unknown, which is an important argument in machine learning. Entropy is a function of the probability p. The greater the probability of an event, the less its uncertainty, that is, it is unknown whether the event will occur or not [4].

And the third no less important parameter is the accuracy metric (accuracy). Accuracy metrics are used in case of unreliability in the sample, that is, representatives of different classes can meet with different probabilities [5]. And to determine the reliability, Keras implements such accuracy metrics as accuracy, binary_accuracy, categorical_accuracy, sparse_categorical_accuracy, top_k_categorical_accuracy, cosine_proximity, sparse_top_k_categorical_accuracy, clone_metric. Accuracy defines the ratio of correctly predicted objects to all other objects. Usually accuracy is calculated by the formula:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}, \quad (1)$$

where TP – True positive (true-positive decision); TN – True negative (true negative decision); FP – False positive (false positive decision); FN – False negative (false-negative decision).

In this study, binary_accuracy was applied, which calculates the coincidence rate of predictions with binary labels, in which y_train should match or have parallel data with y_true. For objective testing of the classifier in problems with images, it is acceptable to use the MSE error metrics [6], but the confusion matrix is more suitable for the text classifier.

To visualize the accuracy metric, the confusion matrix is used, which is usually represented as in Fig. 1.

Each position is calculated with respect to formulas (2)–(5), where n is the number of classes, the class label takes the value +1 (positive class) or –1 (negative class). 4 values are entered, respectively:

$$TP\,(\text{True positive}) = \sum_{i=0}^{n} \left[a(x_i) = +1\right]\left[y_i = +1\right], \quad (2)$$

$$TN\,(\text{True negative}) = \sum_{i=0}^{n} \left[a(x_i) = -1\right]\left[y_i = -1\right], \quad (3)$$

$$FP\,(\text{False positive}) = \sum_{i=0}^{n} \left[a(x_i) = +1\right]\left[y_i = -1\right], \quad (4)$$

$$FN\,(\text{False negative}) = \sum_{i=0}^{n} \left[a(x_i) = -1\right]\left[y_i = +1\right], \quad (5)$$

where $a(x_i)$ – class with elements; y_i – result of the learning algorithm.

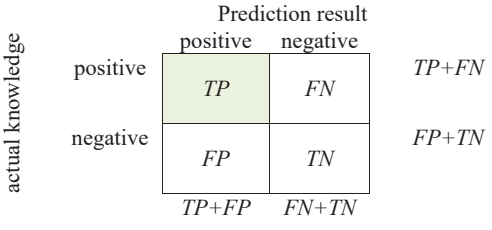

Fig. 1. Matrix of errors

Thus, the TP section shows the sum of the number of classes $a(x_i)$ in which the pair «topic and keywords» have high accuracy and are true both in terms of prediction and the results of the y_i learning algorithm (2). Section TN shows the sum of the number of classes $a(x_i)$ in which the pair «topic and keywords» has false accuracy, both in terms of prediction and according to the results of the y_i learning algorithm (3). Section FP shows the sum of the number of classes $a(x_i)$ in which the pair «topic and keywords» has true prediction accuracy, and false according to the results of the y_i learning algorithm (4). Section FN shows the sum of the number of classes $a(x_i)$ in which the pair «topic and keywords» has false accuracy in prediction, and true accuracy according to the results of the y_i learning algorithm (5). In the FN and FP section, the system shows the errors that were received during learning. Experiments must take into account a certain amount of data. If there is an increase in the data threshold, then fewer false positives and more false negatives are generated. Therefore, one of the curves can rise, and the other – fall. Using this schedule, y_i is possible to choose the optimal threshold value. If there is no such threshold, it is necessary to learn another algorithm.

3. Research results and their discussion

At the time of this study, there is a neural network consisting of a sequence of layers: Embedding (), SpatialDropout1D (), LSTM (dropout, recurrent_dropout), Dense ()), which is a traditional multi-layer feedforward network using back propagation of errors [7]. Previous studies have identified regularization parameters in neural network layers that make up 70 % of the data for the SpatialDropout1D layer, for the hidden dropout and recurrent_dropout layers. As a result of the experiments, a combination of optimal parameters in the compilation of the neural network model was obtained. Here are some examples from the experiments carried out [8]:

1st combination. model.compile (loss='sparse_categorical_crossentropy', optimizer='adam'), the result of which is shown in Fig. 2. This combination shows that the number of classes in shape changes by 1, then the process stops and shows an error.

 ISSN 2664-9969

```
  history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size,validation_split=0.2)#запись результато
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py", line 1263, in fit
  validation_split=validation_split)
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py", line 907, in _standar
_user_data
  exception_prefix='target')
File "C:\Users\Simple\Anaconda3\lib\site-packages\tensorflow\python\keras\engine\training_utils.py", line 191, in st
rdize_input_data
  ' but got array with shape ' + str(data_shape))
lueError: Error when checking target: expected dense to have shape (1,) but got array with shape (19,)
```

Fig. 2. The result of applying the 1st combination of parameters

2nd combination. model.compile (loss='poisson', optimizer='Adadelta', metrics=['accuracy']) – this combination shows the tendency to 0.5 results of calculating errors in the learning and the tendency to 0 of the learning model [9]. And in the learning accuracy, neither the learning nor the learning model changes during learning.

This is inconsistent with the calculation results produced by accuracy (Fig. 3).

3rd combination. model.compile (loss='poisson', optimizer='adam', metrics=['accuracy']) – this combination of parameters shows that the error function diverges after the *n*-th iteration.

This indicates the need for relearning from the place of discrepancy between the learners and the tested data; the accuracy, in turn, does not change during the learning. This means that the result is unreliable in this case (Fig. 4).

4th combination. model.compile (loss='binary_crossentropy', optimizer='SGD', metrics=['cosine_proximity']) – this combination of parameters shows that the error in the learning data is approaching 0 in parallel.

This means that learning proceeds normally without relearning. The accuracy in calculating the loss of error, respectively, increases and tends to unity.

The result is loss (learning errors)=0.1984, cosine_proximity (error accuracy)=0.2239, val_loss (testing error)=0.1985, val_cosine_proximity (testing error accuracy)=0.2235 (Fig. 5).

This is what was required to be obtained as a result of the experiment.

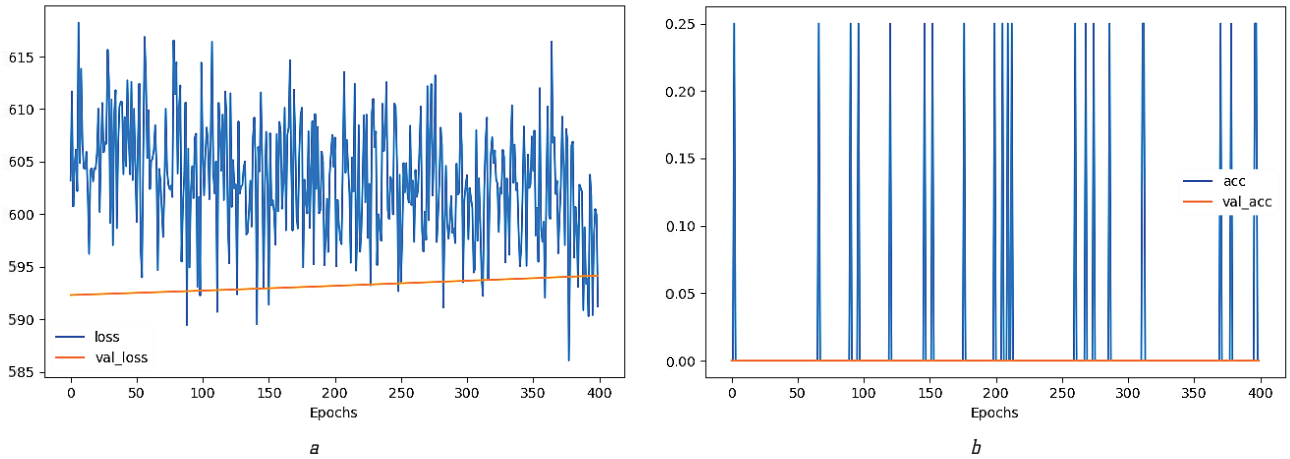

Fig. 3. Result of the 2nd combination: *a* – error (loss, val_los); *b* – accuracy (acc, val_acc)

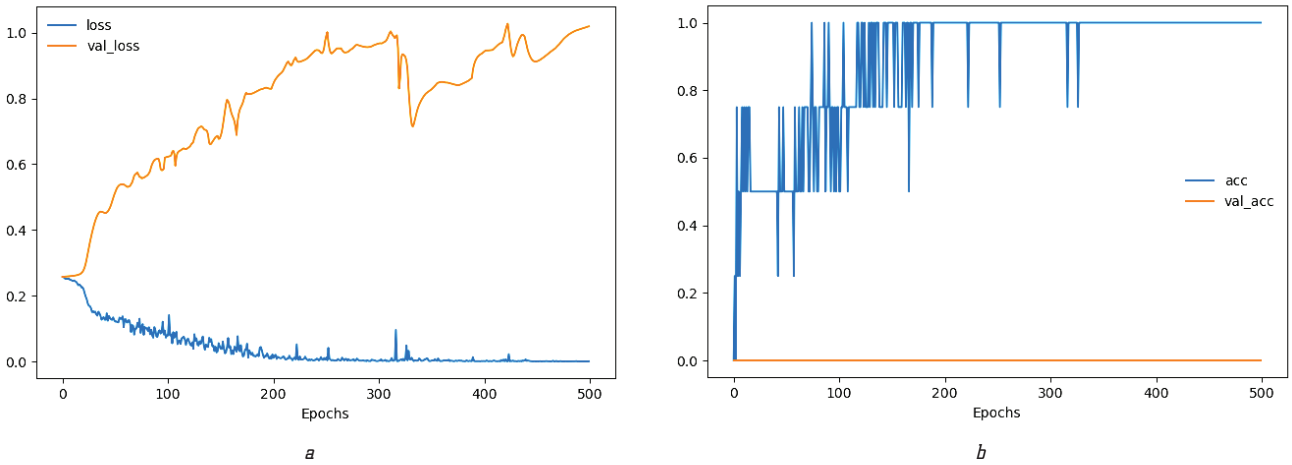

Fig. 4. Result of the 3rd combination: *a* – error (loss, val_los); *b* – accuracy (acc, val_acc)

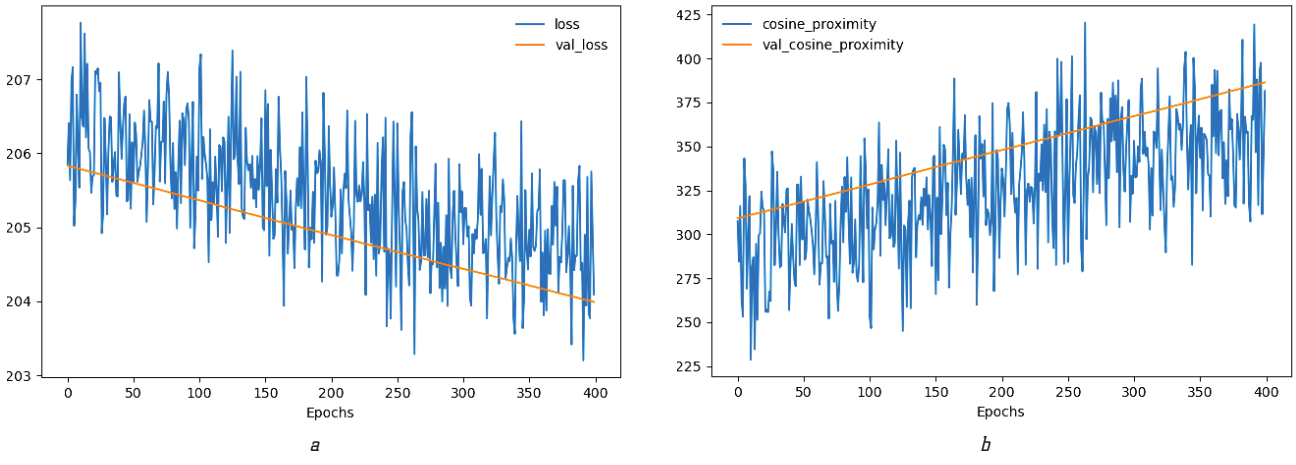

Fig. 5. Result of the 4th combination, specify the values: *a* – error (loss, val_los); *b* – accuracy (acc, val_acc)

4. Conclusions

Thus, there is a sequential composition of the neural network: Embedding (), SpatialDropout1D (0.7)), LSTM (64, dropout=0.7, recurrent_ dropout=0.7)), Dense ()) with the preliminary parameters emb_dim=128, n_most_common_words=8000, batch_size=256, epochs=500. As a result of the experiments carried out using various combinations of arguments in the compilation method of the model, the optimal variant was identified. The optimal combination of compiler arguments when determining the precision of the error loss are the parameters: binary_crossentropy, optimizer=SGD, metrics=cosine_proximity. The research results are applied when conducting deep learning of a neural network, where the sample is text data. The arguments are used in the compile () method of the Keras library for the Python programming language [10].

References

1. Trask, E. (2020). *Glubokoe obuchenie*. Saint Petersburg: Piter, 352.
2. Sholle, F. (2018). *Glubokoe obuchenie na Python*. Saint Petersburg: Piter, 400.
3. Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research, 12*, 2121–2159.
4. Kononiuk, A. E. (2011). I*nformatsiologiia. Obschaia teoriia informatsii. Kniga 3*. Kyiv: Osvita Ukraini, 412.
5. Dzhulli, A., Pal, S. (2018). *Biblioteka Keras – instrument glubokogo obucheniia*. Moscow: DMK Press, 294.
6. Koyuncu, H. (2020). Loss Function Selection in NN based Classifiers: Try-outs with a Novel Method. *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. doi: http://doi.org/10.1109/ecai50035.2020.9223208
7. Hung, C.-C., Song, E., Lan, Y. (2019). Foundation of Deep Machine Learning in Neural Networks. *Image Texture Analysis*, 201–232. doi: http://doi.org/10.1007/978-3-030-13773-1_9
8. *Metriki*. Available at: https://ru-keras.com/metric/
9. Ketkar, N. (2017). *Introduction to Keras. Deep Learning with Python*. Berkeley: Apress. doi: http://doi.org/10.1007/978-1-4842-2766-4_7
10. Chollet, F. (2017). *Deep Learning With Python*. Black & White, 384. Available at: https://github.com/fchollet/deep-learning-with-python-notebooks

Akanova Akerke, *Department of Computer Engineering and Software, S. Seifullin Kazakh Agro Technical University, Nur-Sultan, Kazakhstan, e-mail: akerkegansaj@mail.ru, ORCID: http://orcid.org/0000-0002-7178-2121*

Kaldarova Mira, *Department of Computer Engineering and Software, S. Seifullin Kazakh Agro Technical University, Nur-Sultan, Kazakhstan, e-mail: kmiraj82@mail.ru, ORCID: http://orcid.org/0000-0001-7494-9794*