DL20 GROUP PROJECT - PROJECT PROPOSAL
QUALITY OF LOW RESOURCE TIME SERIES PREDICTION

Emil Faizrakhmanov
emil.faizrakhmanov@tu-ilmenau.de

Aleksandra Fedorova
aleksandra.fedorova@tu-ilmenau.de

Dmitrii Kuziukov
dmitrii.kuziukov@tu-ilmenau.de

Hamza Salaar
hamza.salaar@tu-ilmenau.de

Project Mentor: Johannes Viehweg

December 17, 2020

ABSTRACT

To use Recurrent Neural Networks for time series prediction may require large computing power depending on the accuracy required. This is particularly evident in time series prediction for chaotic systems such as the Lorenz attractor. In this paper, we will consider 4 different neural network architectures: LSTM, GRU, SimpleRNN and Bidirectional LSTM. During the research, we will prepare data for training and optimise neural network's parameters to obtain the highest accuracy of prediction. As a final step of the project, we will compare the obtained results.

1 Introduction

Nowadays time-series prediction is an important task in terms of various field of knowledge and practical applications. One of the complex examples of a time series is a Lorenz attractor. In 1963 MIT meteorologist Edward Lorenz published his paper [1]. Lorenz described a system of nonlinear differential equations that modelled thermally induced fluid convection in the atmosphere. The significance of the Lorenz System was that relatively simple systems could exhibit chaotic behaviour. Now the Lorenz attractor is used to simulate chaotic systems. Prediction of such a system is a challenge because it is difficult to forecast multiple time steps for the system based on chaotic differential equations, so this process is computationally expensive.

In this Group Project, we will use Recurrent Neural Networks (RNN) to predict the time series described above. We will implement several types of RNN architectures. We will vary such parameters as the number of epochs, number of cells in layers, number of layers, and so on. The input to our algorithm is an array of values. The output of our algorithm is one value or sequence of values depends on how many steps ahead we will predict.

2 Related Work

First, we will observe existing solutions described in scientific articles and on web-pages.

The general information about ANN design and optimization is represented in Goodfellow et al. [2]. As a starting point for our algorithm in prediction Lorenz attractor, we will use an approach described in the paper by Madondo et al. [3]. We will also use some features in our algorithm from the approach proposed by Jiménez-Guarneros et al. [4]. Both papers provide a detiled explanation of the Lorenz attractor prediction algorithm. The first one describes one-step ahead prediction, the second one is useful in terms of multi-step ahead prediction.

To implement our algorithm we will use TensorFlow Core for Python [5] and Keras library [6].

3 Dataset and Features

Data represent a Lorenz attractor with the corresponding: σ= 10, r = 28, b = $\frac{8}{3}$. 600.000 values for each x, y and z axis. For training, validation and testing, we have divided the dataset in proportions: 70%, 15%, 5%, respectively. It is planned to normalize the values using MinMaxScaler and compare the neural network training results.

4 Methods

As part of the project, it is planned to use 4 neural networks: LSTM [7], Bidirectional LSTM (an extension of traditional LSTM), GRU [8] and SimpleRNN (Fully-connected RNN where the output is to be fed back to input). The architecture of neural networks consists of several layers: an input layer, LSTM or Bidirectional LSTM or GRU or SimpleRNN cells with the "tanh" activation function. The output layer will be Dense layer with 3 neurons, which will produce the predicted values for each axis. In the process of optimising the neural network, we will select the most appropriate number of layers and cells according to the results of the experiment.

5 Design of Experiment

During the experiment, the error will be measured for all listed before RNN architectures. Root Mean Squared Error (RMSE) will be used as a loss function and Adam as an optimizer. It is planned to carry out 3 measurements of accuracy (losses) for each neural network using: batch_size = 64, 128, 160; epoch = 10, 30, 50. We also provide an early stopping of training if the error does not decrease over several epochs. This is very useful when there are time constraints in training.

6 Contributions

We will divide our future work as follows. First, we will work together to prepare a dataset for training the neural network. Then, we will divide the types of neural network architectures between us. In this way, each will optimize its type of neural network for the specified time series.

References

[1] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20:130–141, 1963.

[2] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[3] Malvern Madondo and T. Gibbons. Learning and modeling chaos using lstm recurrent neural networks. Duluth, Minnesota, USA, 2016. Midwest Instruction and Computing Symposium (MICS).

[4] Pierre Dubois, Thomas Gomez, Laurent Planckaert, and Laurent Perret. Data-driven predictions of the lorenz system. *Physica D: Nonlinear Phenomena*, 408:132495, 2020.

[5] Google Brain team. Tensorflow core v2.3.0. `https://www.tensorflow.org/api_docs/python/tf`.

[6] Keras team. Keras: the python deep learning api. `https://keras.io`.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.