Rules for Selecting Neural Network Architectures for AutoML-GPU Challenge

Abhishek Thakur A.THAKUR@SEARCHMETRICS.COM
Searchmetrics Inc., Berlin, Germany

Abstract

This paper describes the framework used for designing neural networks for the AutoML challenge. The rules for choosing neural network architecture discussed in this paper have been derived after working on many different datasets. The paper also describes how one can easily create and build neural networks with good performance without much effort and with a very little knowledge about the data.

Keywords: machine learning, neural networks, hyper-parameter tuning

1. Introduction

In this paper, we describe the framework we used for selecting neural network architectures for the GPU phase of the AutoML challenge.

The paper is divided into four sections. In section 2, we discuss the framework followed by Section 3 which shows the performance of the framework in the AutoML challenge. Section 4 concludes the paper with future work. References are listed at the end of the paper.

2. Base Framework for Designing Neural Networks

The framework for selecting the neural network architecture for the AutoML-GPU phase is similar to what has been discussed in (Thakur and Krohn-Grimberghe, 2015). Figure 1 provides an overview of the rules used for preprocessing the data.

The very first step in the framework is identification of the task. The task identifier identifies the task of the machine learning problem i.e. whether it is a multi-class classification task, multi-label classification task or a regression problem. The dataset then split to validation and training set depending on the labels. For a multi-class classification a stratified split is used to ensure that the ratio of classes in the validation set is same as that of the training set. Multi-label and regression datasets are split randomly. Task identifier stores the following information about the dataset (provided by the AutoML challenge (Guyon et al., 2015)).

© 2016 A. Thakur.

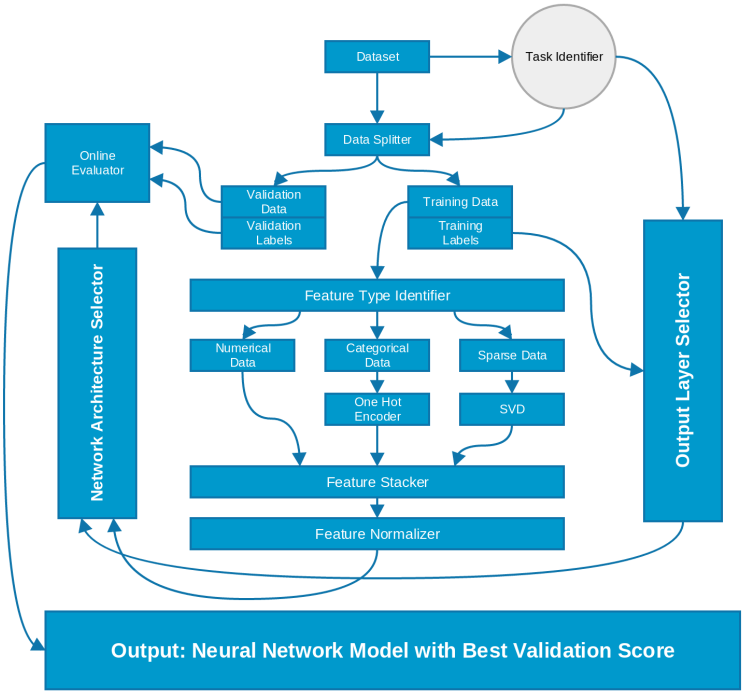

Figure 1: Base framework for training neural networks for AutoML Challenge.

Once we have the validation and training data separated, we move to identification of feature types. For neural networks, this step is simplified when compared to (Thakur and Krohn-Grimberghe, 2015). The numerical features are not touched and sent directly to feature stacker. If categorical features are present, they are one hot encoded and then passed to the feature stacker. In case of text data (after performing tf-idf) or very sparse data, we perform singular value decomposition (SVD). It was found that the appropriate number of components for SVD like between 120 to 180 for datasets in AutoML challenge and this is a number we can fit into GPU memory.

One of the very important steps before feeding data into neural networks is data standardization or normalization. This step is performed by feature normalizer which does scaling of data based on mean and variance, scaling the features between 0 and 1, log scaling and scaling without mean. Datasets scaled using these different methods are then fed to the network architecture selector for selection of the number of layers, activation functions, loss function and optimizer to be used.

The network architecture selector selects the network based on these rules which mostly consisted of sequential models with a dense layer, a batch normalization layer (Ioffe and Szegedy, 2015), a dropout layer (Srivastava et al., 2014) to avoid over-fitting and an activation layer (ReLU (Nair and Hinton, 2010) or PReLU (He et al., 2015)). The output layer selector chooses the appropriate final layer for the neural network. For multi-class classification problems we choose a softmax layer, for multi-label classification tasks a sigmoid layer is chosen and linear activation layer for regression tasks. Similarly, we choose

categorical cross-entropy or multi-class logloss for classification tasks and squared error for regression tasks as the loss function. We used adam optimizer (Kingma and Ba, 2014) for all the datasets.

The most interesting step in network architecture selector is selection of number of layers and their parameters. This is performed by a modified random search. The selector starts with a single layer network with a dense layer of 120-500 neurons, batch normalization, PReLU activation and a dropout of 10-20%. The performance is noted and an extra layer with the same configuration is added to the network. With the online evaluator, we measure the performance of the network with these two different configurations. The number of neurons is then increased in both layers to 1200-1500 and performance is noted again. If the network does not perform better than the previous configuration, the dropout is increased to 40-50%. A big network with 8000-10000 neurons is then chosen with the same dropout and results are recorded again. If everything fails or if the network overfits, the dropout is increased to a very high value ranging from 60% to 80% and performance is recorded again. The network selector keeps adding layers, changes number of neurons per layer, changes dropout, checks if batch normalization improves performance and records the performance of the network. Since all the values are chosen from a bucket of parameters with fixed ranges, we don't go into endless loop.

3. Results on AutoML-GPU Datasets

The datasets in the final phase of AutoML-GPU challenge were named as: evita, flora, helena, tania and yolanda. The scores obtained in the final phase of the AutoML GPU track by using the framework described in Section 2 are shown in Table 1.

Dataset	**Evita**	**Flora**	**Helena**	**Tania**	**Yolanda**
Evaluation Metric	**AUC**	**A Metric**	**BAC**	**PAC**	**R2**
Scores	0.5694	0.5001	0.2381	0.7617	0.3870

Table 1: Scores obtained by the described framework in AutoML GPU track

The resulting networks from the framework rules secured first place in the GPU phase of the competition.

4. Conclusion and Future Work

This paper describes a framework for selection of neural network architectures with preprocessing of the datasets.

Although the current framework worked good in the AutoML challenge, it still has a lot of scope for improvement. The framework can be improved by addition of more preprocessing steps, a better way of selection of parameters for the neural network models and addition of models like LSTMs and convolutional neural networks.

References

I. Guyon, K. Bennett, G. Cawley, H. J. Escalante, S. Escalera, T. Kam Ho, N. Macià, B. Ray, M. Saeed, A. Statnikov, and E. Viegas. Design of the 2015 ChaLearn AutoML challenge. In *Proc. of IJCNN*, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015. URL http://arxiv.org/abs/1502.01852.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010. URL http://www.icml2010.org/papers/432.pdf.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

A. Thakur and A. Krohn-Grimberghe. Autocompete: A framework for machine learning competitions. In *AutoML Workshop, International Conference on Machine Learning 2015*, 2015.