Project 1.1: solving FizzBuzz

Volodymyr Liunda
Department of Computer Science and Engineering
Buffalo University
Buffalo, NY 14260
vliunda@buffalo.edu

Abstract

FizzBuzz is a popular basic programming interview problem, which is usually implemented using if-else statements. As an alternative we propose a machine learning based solution which uses a feedforward neural network to solve the same task. Experiments concluded on generated data show that with right data preprocessing this approach achieves high accuracy and can be used to solve similar tasks.

1 Introduction

We consider the multiclass classification task of predicting a desired label ***l*** given a number ***n***, where $n \in N$. Specifically all numbers divisible by 3 are assigned label 'Fizz', divisible by 5 'Buzz', divisible by both 3 and 5 - 'FizzBuzz'. Other digits are assigned label 'Other'. Classification is performed by multilayer perceptron which is a powerful model that learns how to use parameterized hidden layers to classify numerical samples, given their binary representation. Some model can be trained to classify digits into different categories. General modelling framework can be used for much more complicated tasks. In this report we make the following contributions:

(a) Show that a learning algorithm can perform FizzBuzz classification task.
(b) Explore the influence of model hyperparameters on model performance and propose best set of parameters for given task.
(c) Show and explain model limitations.

2 Method

It can be seen from [1],[2] that multilayer perceptrons can be used to solve classification problems and that it can distinguish data that is not linearly separable [3]. However, if we use number ***n*** as a single network input, the model performs poorly as such parameters as divisibility cannot be easily extracted from given representation. Single most popular way to represent digits in computer systems is binary. It is a base-2 number system that uses two mutually exclusive states to represent numerical information. It represents integer ***n*** as an array ***b={b₀, b₁ .. bₘ}*** such as:

$$n = \sum_{i=1}^{m} 2^{i} * b_i$$

This simplifies pattern discovery as now we have a vector of binary features. Due to limitations of neural networks we must limit the number ***m*** of bites representing a number. For the purpose of this experiment we set *m=10* to represent numbers 0 to 1023. If we test this algorithm on larger numbers the accuracy will fall as we will get identical representations for different numbers such as 0 and 1024.

3 Experiments

3.1 Optimal hyperparameters

To find the best model architecture and learning parameters we conducted bayesian hyperparameter search using hyperopt library and measured model performance using such metrics as categorical cross entropy and accuracy of classification. Search domain is represented in Table 1.

Table 1: Hyperparameter search space

Hyperparameter	**Values**
Activation	tanh, relu, elu, selu
Num. of hidden units	1-3
Num. hidden units 1st layer	*N*(256,10)
Num. hidden units 2nd layer	*N*(128,10)
Num. hidden units 3rd layer	*N*(64,5)
Dropout rate	*U(*0,0.9*)*
Learning rare	*logUniform(log(*0.001), log(0.1*))*

Obtained optimal parameters can be found in Table 2.

Table 2: Optimized hyperparameters

Hyperparameter	**Values**
Activation	tanh
Num. of hidden units	3
Num. hidden units 1st layer	*245*
Num. hidden units 2nd layer	*154*
Num. hidden units 3rd layer	*67*
Dropout rate	*0.339*
Learning rare	*0.00815*

3.2 Dropout rate influence

To check how hyperparameters influence model performance we've also measured accuracy for 4 categories after training with 20 different dropout rates ranging from 0 to 0.9. To reduce influence of stochastic factors we used the same set of weights to initialize every model. The results are shown on Figure 1.

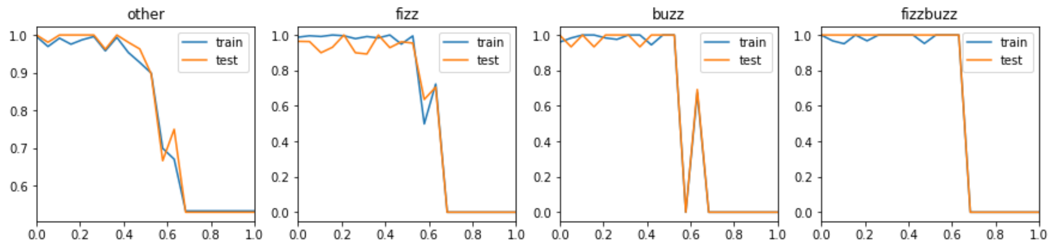

Figure 1: Dropout rate influence on model learning performance

One can clearly see that having more than 50% nodes dropped reduces learning performance drastically, which signifies that we should not reduce the number of layers or nodes per layers. Another finding is that the model is not overfitting even with small dropout rate values.

3.3 Optimal learning algorithm

We also checked how choosing a learning algorithm affects learning performance. Four algorithms were chosen for the test and we used same learning rate for all of them. The results of the experiment can be seen in Table 3.

Table 3: Optimization algorithm comparison

Optimization algorithm	**Test accuracy**	**Train accuracy**	**Num. epochs**
SGD	0.133	0.134	108
RMSprop	1	0.986	808
Adam	*0.981*	0.987	*858*
Nadam	*0.995*	*0.971*	*690*

RMSprop, Adam and Nadam perform almost at the same level, though RMSprop is a clear winner as it converges faster than Adam and perform better than Nadam on both train and test sets. SGD performs poorly both on train and test sets. Further investigation showed that SGD gets stuck in a local optima as validation loss is decreasing during training, but validation accuracy does not improve. Increasing learning rate does not help as optimization does not converge and learning ends due to early stopping, neither helps changing activation function from *tanh* to a more robust to gradient explosion *relu*. Adding momentum and learning rate decay makes no difference as well. To further investigate this issue we plan to check gradients during learning and compare them with gradients from different algorithms.

3 Conclusion

In this work we showed that traditional problems like FizzBuzz can be solved using learning algorithms that find hidden patterns using training data. This approach is not completely accurate, but allows us to train the same network architecture on different datasets to perform different tasks. Also learning algorithms do not require the programmer to know the logic behind the pattern in provided data. Therefore such algorithms are efficient when applied to problems that are hard to solve using traditional rule-based logic.

References

[1] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961

[2] Rumelhart, David E., Geoffrey E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". David E. Rumelhart, James L. McClelland, and the PDP research group. (editors), Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundation. MIT Press, 1986.

[3] Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function *Mathematics of Control, Signals, and Systems*, 2(4), 303–314.