CSE 574: Introduction to Machine Learning

Amlan Gupta
#50288686
amlangup@buffalo.edu

Abstract

In this project, we are going to implement fizzbuzz game using both traditional logic-based implantation and machine learning based approach. The machine learning implementation will be powered by open source neural network library written in python which is an abstraction layer on top of TensorFlow. We will exhibit the contrast between two approaches as well as how changes in the environment can affect the decisions of machine learning implementation.

1 General Concepts

1.1 What is Machine Learning?

Machine learning is a technique provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

In this context, we will feed the system sample data to learn from it. A CSV file from 101 to 1000 with proper labels will be created, which will be read by the system and a prediction model will be generated. Using this, the system will try to guess the right answer for the numbers.

1.2 What is Neural Network?

Neural Network is loosely modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales. A neural net consists of thousands or even millions of simple processing nodes(neurons) that are densely interconnected. Most of today's neural nets are organized into layers of nodes, and they're "feed-forward," meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the layer above it, to which it sends data.

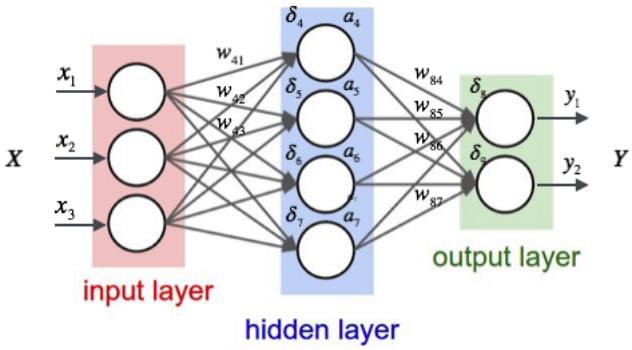


Figure 1: Neural Network [1]

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

The nodes or neurons of the input layer is passive. All they do is pass the data received from input to the next layer. Nodes in Hidden layer and output layer are active.

The values entering a hidden node are multiplied by weights, a set of predetermined numbers stored in the program. The weighted inputs are then added to produce a single number.

Neural networks can have any number of layers, and any number of nodes per layer. Most applications use the three-layer structure with a maximum of a few hundred input nodes.

2 Solution walk-through and their underlying concepts

Problem Statement

The problem statement is based on a children's counting game called 'Fizzbuzz'. In this task an integer divisible by 3 is printed as Fizz, and integer divisible by 5 is printed as Buzz. An integer divisible by both 3 and 5 is printed as FizzBuzz. If an integer is not divisible by 3 or 5 or 15, it simply prints the input as output

The program will be tested on how well they perform in converting integers from 1 to 100 to the FizzBuzz labels.

2.1 Generating sample data to be used by NN

```
def createInputCSV(start,end,filename):

    inputData   = []
    outputData  = []

    for i in range(start,end):
        inputData.append(i)
        outputData.append(fizzbuzz(i))


    dataset = {}
    dataset["input"]  = inputData
    dataset["label"] = outputData


    pd.DataFrame(dataset).to_csv(filename)

    print(filename, "Created!")


createInputCSV(101,1001,'training.csv')
createInputCSV(1,101,'testing.csv')
```

As instructed by problem Statement, we will perform testing by using values from 1 to 100.

So we are creating to data set: one for training the neural network (value from 101 to 1000) and one testing set. The expected output are included in the data set to both train and test the accuracy of the neural network.

Note: fuzzbuzz() function is the logical implementation of this problem. We are using this to generate sample data.

2.2 Create Model

```
input_size = 10
drop_out = 0.2

def get_model():

    model = Sequential()

    model.add(Dense(128, input_dim=input_size))
    model.add(Activation('relu'))

    model.add(Dropout(drop_out))

    model.add(Dense(328))
    model.add(Activation('relu'))

    model.add(Dropout(drop_out))

    model.add(Dense(512))
    model.add(Activation('relu'))

    model.add(Dropout(drop_out))

    model.add(Dense(4))
    model.add(Activation('softmax'))

    model.summary()
    model.compile(optimizer='rmsprop',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model



model = get_model()
```

Model is the blueprint of neural network. All the layers, their properties, activations, optimizers, used loss function etc are stored in the model itself.

2.2.1 Sequential Model

Keras has support for 2 types of models: Sequential and Functional API. In sequential model, we can add layers one-by-one. It has one way data flow, means, first level's output will be used as second layer's input and so on.

Functional Model has more flexibility and not restricted by the previous and next layer. But it can be used in comparatively more complex problem.

2.2.2 Hidden Layer

Through input layer training data inserts into the neural network. The neural network puts random weights on input and send it to the next hidden/dense layer. The next layer applies a series of functions

to the data. Most frequently, these functions each compute a linear transformation of the previous layer, followed by a squashing non-linearity. So the roles of the different layers could depend on what functions are being computed.

2.2.3 Activation

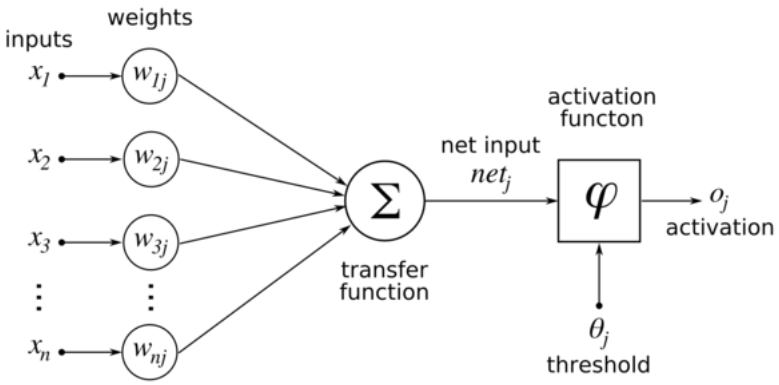


Figure 2: Activation Function [2]

Activation functions are an extremely important feature of the artificial neural networks. They basically decide whether a neuron should be activated or not. Activation is used to add non-linearity in the neural network. If we have three dense layer without activation, then it is nothing but a linear function, albeit more complex one. Some of the popular activation functions are: Relu, sigmoid, tanh, leaky relu, softmax etc.

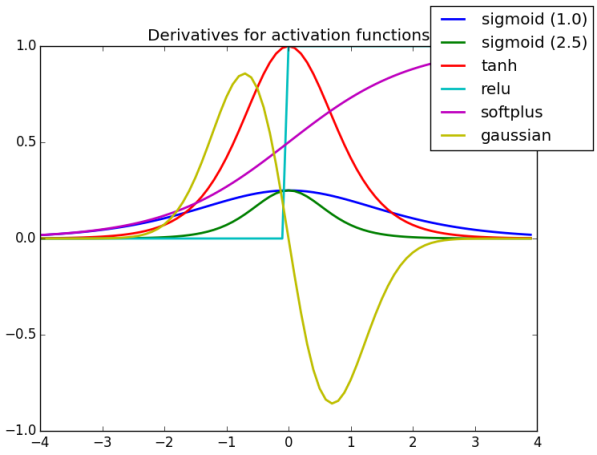
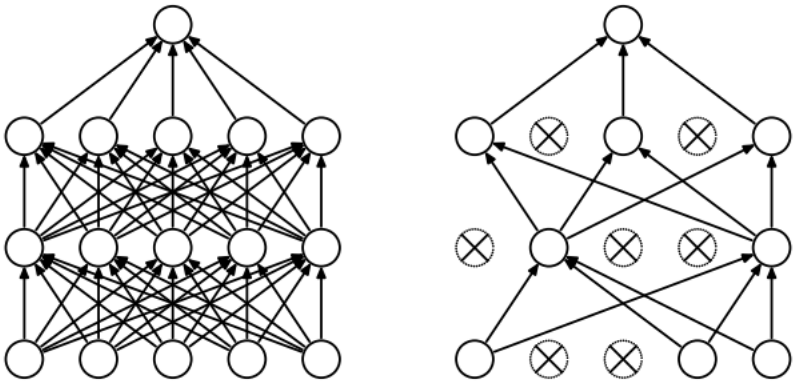


Figure 3: Derivative of different activation functions [3]

2.2.4 Dropout



(a) Standard Neural Net (b) After applying dropout.

Figure 4: Dropout [4]

Dropout is a technique used in neural network to reduce over-fitting. With bigger neural networks, once a few dense layers are introduced and it becomes so complex that, the learning rate becomes stagnant and process becomes slow. Dropout is used to minimize that situation.

2.2.5 Optimizer

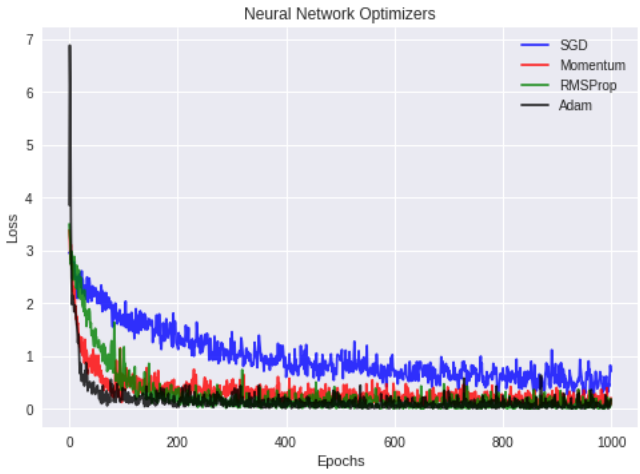


Figure 5: training loss vs iterations for different optimization algorithms

Optimization algorithms helps us to minimize (or maximize) error function which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model. Some of the popular optimizers are SGD, RMSProp, AdaGrad, Adam etc.

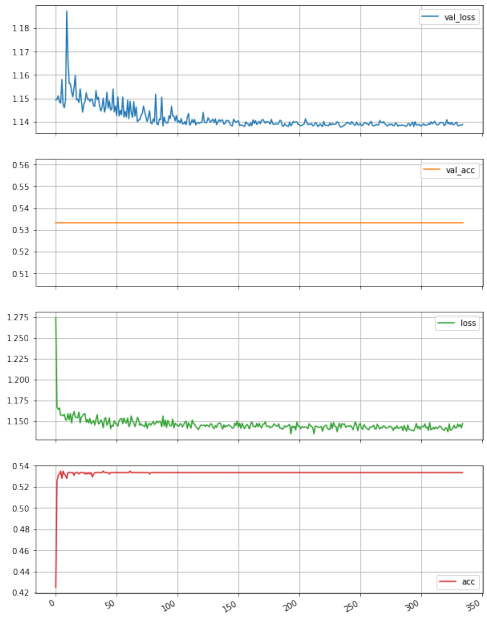
2.3 Learning & Testing

Once the training data is passed into the neural network, network feeds the data to the above mentioned model. After one epoch the error will typically get passed backwards through the network and the feedback algorithm will individually increase or decrease those weights by some proportion to the error. The network will repeatedly iterate by passing forward, measuring the output response, then

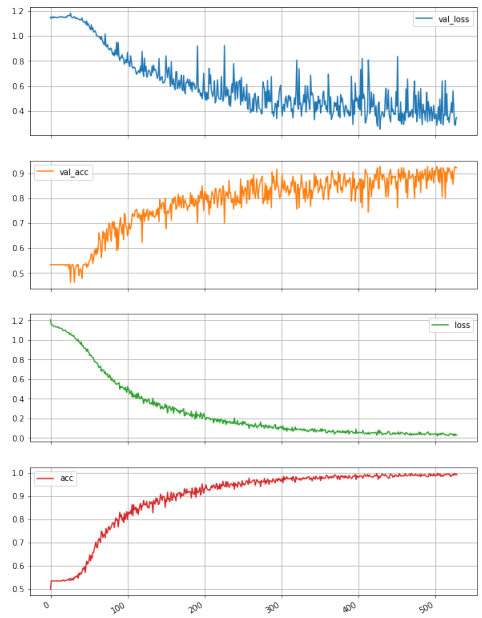
updating (passing backwards weight adjustments) and correcting the weights until some satisfactory error level is reached.

3 Hyperparameter Optimization

a Changing the first activation from 'relu' to 'tanh' we get the accuracy of 53%:



b Adding two new layers of dense layer of 312 and 512 nodes and two relu activation, we get the accuracy of 89%:



4 Conclusion

The prediction was most accurate (95%) with the following combination:

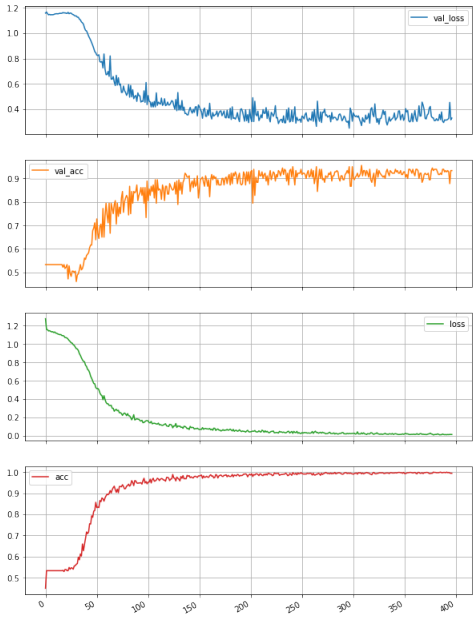
first dense layer: 512 nodes

first activation: relu

Second dense layer: 256 nodes

Second activation: relu

Optimizer used: adam



References

[1] Venelin Valkov. *Creating a Neural Network from Scratch—TensorFlow for Hackers (Part IV)*. Meduim, 2017.

[2] Chrislb. *Diagram of an artificial neuron*. wikipedia, 2005.

[3] Sagar Sharma. *Activation Functions: Neural Networks*. medium, 2014.

[4] Alex Krizhevsky Ilya Sutskever Ruslan Salakhutdinov Nitish Srivastava, Geoffrey Hinton. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. wikipedia, 2014.

[5] andrew Ng. Machine learning. Coursera, 2012.

[6] Keras. Keras documentation. keras.io, 2015.