849 DATA SCIENTIST HOMEWORK PROBLEM

SIAMAK RABIENIA

1. Introduction

Given the dataset "dataset.npy", our task is Genome Sequence Classification where we use the Recurrent Neural Network (RNN). Since our dataset consists of Genome Sequences and our goal is to find the pattern in each given sequence, we specifically utilize the Long Short-Term Memory RNN in Deep Learning which allows us to not only process single data points, but also entire sequences of data. Another example, where LSTM is widely used, is in Natrual Language Processing to learn and predict the word patterns.

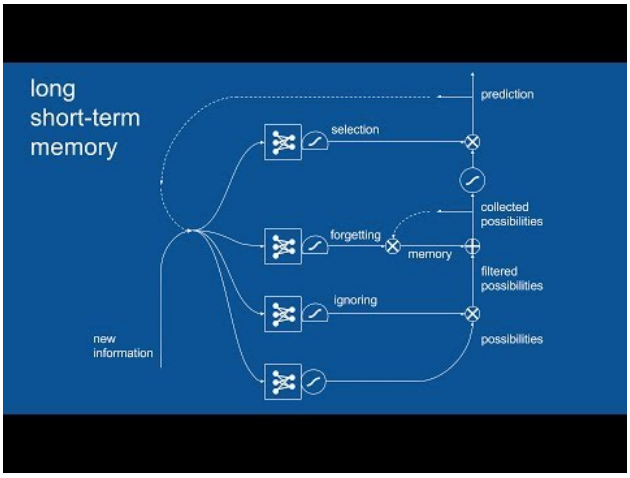


Figure 1. Long Short-Term Memory RNN Structure

2. Recurrent Neural Network

The size of genome sequences in dataset are variable. Therefore, we perform padding to fix the size to 190. If the size of the genome is less that 190, our algorithm adds zeros to the sequence to make of size 190.

The Structure. The structure of our neural network is as follows: the input layer is our genome sequence. The hidden layers consist of four different layers: An embedding layer, which typically project the input space (sequence of size 190) to the lower dimension spaces, as well as two LSTM layers followed by a $\lambda-$layer. At the end we have the output layer, see Figure 2. By noting i the $i-$ th layer of the network and j the $j-$th hidden unit of the layer, we have:

$$z_j^{[i]} = {w_j^{[i]}}^T x + b_j^{[i]},$$

where we w, b, z are the weight, bias, and output respectively.

Date: September 30, 2019.

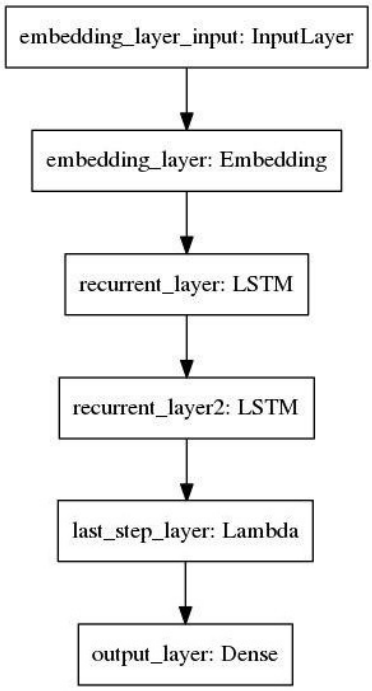


FIGURE 2. LSTM RNN Layers

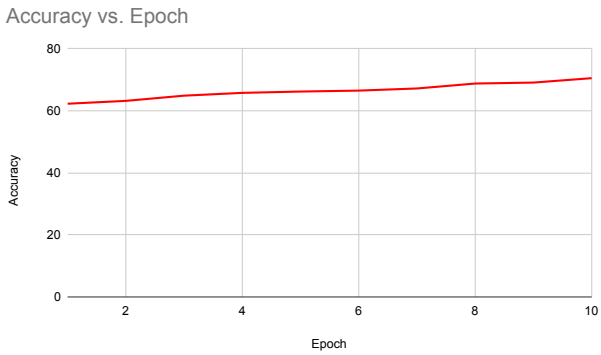
For our LSTM RNN, the Sigmoid function which converts the estimated probabilities to values between 0 and 1

$$f(z) = \frac{1}{1 + e^{-z}}$$

has been used as an activation function at the end of a hidden unit to introduce non-linear complexities to the model. Since we employed the Sigmoid function to activates the neurons, we use Cross-Entropy Loss (or Log Loss) function defined as follows:

$$L(z, y) = -\big[y \log(z) + (1 - y) log(1 - z)\big].$$

To train the neural network we use the Adam's optimizer which is a Dynamic Stochastic Gradient Descent algorithm where it updates the network weights iteratively based on the training data as oppose to the regular Stochastic Gradient Descent methods where the learning rate will not be changed during the training procedure. We also use the Drop-Out set with 18% portion which acts like a regularizer in our neural network structure. We split the data to 80%-20% train-test and perform the 5-fold cross-validation to avoid over-fitting issues and discover the hyper-parameters (like epoch size, neuron size, etc.). The neuron size is 60 and for the choice of batch size, we tested the size of 300, 400, and 500 for training of the dataset and selected the size of 400 as it resulted to the highest accuracy and lowest error on training set. Our results are given by Figure 3.


Accuracy vs. Epoch
80
60
40
20
0
Accuracy
2
4
6
8
10
Epoch


Train Loss and Test Loss
Train Loss
Test Loss
1.25
1
0.75
0.5
0.25
0
2
4
6
8
10
Epoch