IIT Bombay

CS 744

Design and Engineering of Computing Systems

Key-value Store

Author:
Atul Sahay

Roll No:
18305R003

March 17, 2019

Contents

1 Test Environmemt Setup

The initial Environment setup that I followed during the conduct of the analysis study of the performance measure of the system is, I have used an UBUNTU 18.04 O.S based on debian linux distribution. The whole program is divided into two sets of **load-generator** and **server**. Both of which run on the same system having the O.S as described above. The system is having RAM space of 4GB while the processor is intel core i5 processor with clock speed of 2.5GHz. The server is made to run on 2 cores using the **taskset** command, while load-generator is made to run on the other two left cores of the system.

For reducing the deviation from the analysis report all the other processes are cleaned and only these two processes are made to run.

2 Task 2

1. α : Learning Rate
2. *f(x)* : Activation Function: Sigmoid

$$g(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

$$g'(x) = g(x)(1 - g(x)) \tag{2}$$

3. Accuracy : On test data , as reported on kaggle.
4. Batch Gradient Descent : Batch Size = 100.

No of hidden layers	Neurons	α	Accuracy
1	100	0.001	–
4	100, 150,200,250	0.001	–

3 Task 3

3.1 Setting Learning Rate: α

3.1.1 How to set learning rate.

1. Batch Size = 100.
2. Feed Forward and Backpropagate on first batch.
3. Calculate Validation loss for the first time and append in a list "loss".
4. Feed Forward on next batch and do backpropagation.
5. Calculate validation loss and check if it is less than minimum validation loss's in the past i.e. minimum of "loss" list.
6. If abs(current validation loss - minimum past validation loss) is less than 1(one) then
   learning rate = learning rate / (abs(current validation loss - minimum past validation loss) + 1)
7. Append the current loss in list "loss"

- With starting learning rate = 0.01 , the observations lead it to a decreased learning rate of 0.005.
- With starting learning rate = 0.001 , the observations lead it to a decreased learning rate of 0.0005. These are approximate observations stater here. For actual reading, printing learning rate in code will work out.

3.1.2 Observations: Decay Learning Rate

Learning rate	Num/size of hidden layers	λ	Training loss	Validation loss	Traini
0.001	1(100)	1	0.3524962414679528	0.3497653162784684	0.7447
0.001	1(100)	3	0.3541047341403841	0.3513913070163647	0.7443

3.2 L2 Regularization and Observations

Learning rate	Num/size of hidden layers	λ	Training loss	Validation loss	Tra
0.001	1(100)	1	0.3472218908682681	0.3453361048217388	0.74
0.001	2(100,100)	1	0.3416874314959269	0.34092385621741206	0.74
0.001	3(50,100, 50)	1	0.34857327112390263	0.34629619813079426	0.74
0.001	4(100,150,200)	1	0.34150086948685365	0.33819506097124125	0.74
0.001	4(100,200, 300, 400)	1	0.34429085456528596	0.3371596941958156	0.74

4 Different Activation Functions

4.1 tanh

1. α : Learning Rate
2. *f(x)* : Activation Function: tanh

$$f(x) = tanh(x) = 2 * g(2x) - 1 \tag{3}$$

$$g(x) = \frac{e^x}{1 + e^x} \tag{4}$$

$$f'(x) = 1 - tanh^2(x) \tag{5}$$

3. Accuracy : On test data , as reported on kaggle.
4. Batch Gradient Descent : Batch Size = 100

No of hidden layers	Neurons	α	Accuracy
1	100	0.001	–
4	100, 150,200,250	0.001	–

4.2 ReLU

1. α : Learning Rate
2. *f(x)* : Activation Function: ReLU

$$f'(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \ . \end{cases} \tag{6}$$

$$f'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x <= 0 \ . \end{cases} \tag{7}$$

3. Accuracy : On test data , as reported on kaggle.
4. Batch Gradient Descent : Batch Size = 100

No of hidden layers	Neurons	α	Accuracy
1	100	0.001	–
4	100, 150,200,250	0.001	–

5 Optimization

1. α : Learning Rate
2. *f(x)* : Activation Function: Sigmoid

$$g(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

$$g'(x) = g(x)(1 - g(x)) \tag{9}$$

3. Accuracy : On test data , as reported on kaggle.
4. Batch Gradient Descent : Batch Size = 100.
5. Number of hidden layers: 4 (100, 150, 200, 250).
6. Number of iterations: 1000.

One iterations equals 288 batches of size 100 each.