Realization Of Identity Function Using Two Node Neural Network

A simple function, the Identity Function y = x, is used to explore Stochastic, Mini-Batch and Batch processing with Gradient Descent(GD), Gradient Descent With Momentum(GDM), RMSProp and Adam optimizer.

# **Network**

The predicted output, , is calculated given the input, , along with weight, , and bias, . The predicted outputis then compared against the actual output using Mean Squared Error Loss, , which gives a measure of how far apart the two outputs are. The gradient of the loss with respect to weight and bias, and respectively, helps in moving in the right direction so as to minimize the loss.

Actual Output :

Predicted Output :

The general flow is defined as below:

Forward Propagation **:**

Loss **:**

Gradients **:**

**:**

Updates :

:

The data processing method, Stochastic, Mini-Batch or Batch processing, determines how often is the weight and bias updated. The optimizer method, GD, GDM, RMSProp or Adam, determines how to use the gradient information.

# **Input Sample Processing**

## **Stochastic Processing**

The weight and bias is updated for every input sample.

## **Mini-Batch Processing**

The gradient is averaged over M(Mini-Batch Size) samples. The weight and bias is updated after every M samples.

**,**

**,**

## **Batch Processing**

**,**

**,**

The gradient is averaged over all the N samples. The weight and bias is updated after N samples.

# **Optimizer**

and will be used in the below equations and they can be either Stochastic, Mini-Batch or Batch processing output.

## **Gradient Descent**

The gradient is used as it is to update the weight and bias.

## **Gradient Descent With Momentum**

The exponential moving average of the gradient is used to update the weight and bias.

## **Root Mean Square Prop**

The exponential moving average of the squared gradient is used to update the weight and bias.

## **Adam**

The exponential moving average of both the gradient and squared gradient is used to update the weight and bias.

# **Performance**

As this is an Identity Function, we know that that the weight should be 1 and bias 0. The training is stopped when both weight and bias reach within 10E-6 of the expected value.

## **Stochastic Processing with Gradient Descent**

With learning rate, set to 0.001, the weight and bias explodes(obtain high magnitude) immediately thereby making the network useless.

Within 1815 samples, the weight and bias approach infinity. Below graph shows the high oscillation in weight in just ~50 samples

Lowering to 0.0001 prevents the weights from exploding and we get a usable network.

The training stops after 131717 samples taking ms.

## **Stochastic Processing with Gradient Descent with Momentum**

is 0.0001

The training stops after 15548 samples taking ms.

## **Stochastic Processing with RMS**

is lowered to 0.1

The training stops after 843 samples taking ms.

## **Stochastic Processing with Adam**

is 0.1

The training stops after 590 samples taking ms.

## **Mini-Batch Processing with Gradient Descent**

## **Mini-Batch Processing with Gradient Descent with Momentum**

is 0.001

## **Mini-Batch Processing with RMS**

is 0.1

## **Mini-Batch Processing with Adam**

is 0.1

## **Batch Processing with Gradient Descent**

is 0.0001

## **Batch Processing with Gradient Descent with Momentum**

## **Batch Processing with RMS**

is 0.000001

Time taken is 746926776 microseconds

## **Batch Processing with Adam**

is 0.000001

Time taken is 738568845 microseconds = 12.30948075 mins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **GD** | **GDM** | **RMSProp** | **Adam** |
| **Stochastic** |  |  |  |  |
| **Mini-Batch** |  |  |  |  |
| **Batch** |  |  |  |  |

#define EPOCHS (5000)

#define LEARNING\_RATE (0.01)

Time = microseconds

#define EPOCHS (5000)

#define LEARNING\_RATE (0.0001)

Time = microseconds

#define EPOCHS (10000)

#define LEARNING\_RATE (0.00001)

Time = 17829730 microseconds

#define EPOCHS (10000)

#define LEARNING\_RATE (0.0001)

Time = 20217466 microseconds

#define EPOCHS (20000)

#define LEARNING\_RATE (0.00001)

Time = 43081733 microseconds

#define EPOCHS (50000)

#define LEARNING\_RATE (0.00001)

Time = 105537851 microseconds

#define EPOCHS (80000)

#define LEARNING\_RATE (0.00001)

Time = 174059978 microseconds

#define EPOCHS (100000)

#define LEARNING\_RATE (0.000001)

Time = 189396820 microseconds

#define EPOCHS (200000)

#define LEARNING\_RATE (0.000001)

Time = 410279056 microseconds

#define EPOCHS (400000)

#define LEARNING\_RATE (0.000001)

Time = 738568845 microseconds

Do timing analysis in release mode